XUEDONG HUANG | ALEX ACERO | HSIAO-WUEN HON

SPOKEN LANGUAGE PROCESSING

A Guide to Theory, Algorithm, and System Development



Foreword by Dr. Raj Reddy
Carnegie Mellon University

Petitioner has added romanettes to pages (i) - (iii). Otherwise, it leaves the original page numbering.

MICROSOFT CORP.
EXHIBIT 1011

Spoken Language Processing

A Guide to Theory, Algorithm, and System Development

Xuedong Huang
Alex Acero
Hsiao-Wuen Hon

Microsoft Research



Prentice Hall PTR
Upper Saddle River, New Jersey 07458
www.phptr.com

Library of Congress Cataloging-in-Publication Data

Huang, Xuedong.
Spoken language processing: a guide to theory, algorithm, and system development/
Xuedong Huang, Alex Acero, Hsiao-Wuen Hon.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-022616-5
1. Natural language processing (Computer science) I. Acero, Alex. II. Hon,
Hsiao-Wuen. III. Title.

QA76.9.N38 H83 2001
006.3'5—dc21 00-050196

Editorial/production supervision: *Jane Bonnell*
Cover design director: *Jerry Votta*
Cover design: *Anthony Gemmellaro*
Manufacturing buyer: *Maura Zaldivar*
Development editor: *Russ Hall*
Acquisitions editor: *Tim Moore*
Editorial assistant: *Allyson Kloss*
Marketing manager: *Debby van Dijk*



© 2001 by Prentice Hall PTR
Prentice-Hall, Inc.
Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale.
The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact Corporate Sales Department, Phone: 800-382-3419; FAX: 201-236-7141;
E-mail: corpsales@prenhall.com
Or write: Prentice Hall PTR, Corporate Sales Dept., One Lake Street, Upper Saddle River, NJ 07458.

Company and product names mentioned herein are the trademarks or registered trademarks of their respective owners.

All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.

Printed in the United States of America

ISBN 0-13-022616-5

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte. Ltd.
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Contents

FOREWORD xxi

PREFACE xxv

1. INTRODUCTION 1

1.1. MOTIVATIONS 2
1.1.1. Spoken Language Interface 2
1.1.2. Speech-to-Speech Translation 3
1.1.3. Knowledge Partners 3
1.2. SPOKEN LANGUAGE SYSTEM ARCHITECTURE 4
1.2.1. Automatic Speech Recognition 4
1.2.2. Text-to-Speech Conversion 6
1.2.3. Spoken Language Understanding 7
1.3. BOOK ORGANIZATION 8
1.3.1. Part I: Fundamental Theory 9
1.3.2. Part II: Speech Processing 9
1.3.3. Part III: Speech Recognition 9
1.3.4. Part IV: Text-to-Speech Systems 10
1.3.5. Part V: Spoken Language Systems 10
1.4. TARGET AUDIENCES 10
1.5. HISTORICAL PERSPECTIVE AND FURTHER READING 11

PART I: FUNDAMENTAL THEORY

2. SPOKEN LANGUAGE STRUCTURE 19

2.1. SOUND AND HUMAN SPEECH SYSTEMS 21
2.1.1. Sound 21
2.1.2. Speech Production 24
2.1.3. Speech Perception 29

2.2. PHONETICS AND PHONOLOGY 36
2.2.1. *Phonemes* 36
2.2.2. *The Allophone: Sound and Context* 47
2.2.3. *Speech Rate and Coarticulation* 49
2.3. SYLLABLES AND WORDS 51
2.3.1. *Syllables* 51
2.3.2. *Words* 53
2.4. SYNTAX AND SEMANTICS 58
2.4.1. *Syntactic Constituents* 58
2.4.2. *Semantic Roles* 63
2.4.3. *Lexical Semantics* 64
2.4.4. *Logical Form* 67
2.5. HISTORICAL PERSPECTIVE AND FURTHER READING 68

3. PROBABILITY, STATISTICS, AND INFORMATION THEORY 73
3.1. PROBABILITY THEORY 74
3.1.1. *Conditional Probability and Bayes' Rule* 75
3.1.2. *Random Variables* 77
3.1.3. *Mean and Variance* 79
3.1.4. *Covariance and Correlation* 82
3.1.5. *Random Vectors and Multivariate Distributions* 83
3.1.6. *Some Useful Distributions* 85
3.1.7. *Gaussian Distributions* 92
3.2. ESTIMATION THEORY 98
3.2.1. *Minimum/Least Mean Squared Error Estimation* 99
3.2.2. *Maximum Likelihood Estimation* 104
3.2.3. *Bayesian Estimation and MAP Estimation* 107
3.3. SIGNIFICANCE TESTING 113
3.3.1. *Level of Significance* 114
3.3.2. *Normal Test (Z-Test)* 115
3.3.3. *χ^2 Goodness-of-Fit Test* 116
3.3.4. *Matched-Pairs Test* 118
3.4. INFORMATION THEORY 120
3.4.1. *Entropy* 120
3.4.2. *Conditional Entropy* 123
3.4.3. *The Source Coding Theorem* 124
3.4.4. *Mutual Information and Channel Coding* 126
3.5. HISTORICAL PERSPECTIVE AND FURTHER READING 128

4. PATTERN RECOGNITION 133
4.1. BAYES' DECISION THEORY 134
4.1.1. *Minimum-Error-Rate Decision Rules* 135

4.1.2. *Discriminant Functions* 138
4.2. HOW TO CONSTRUCT CLASSIFIERS 140
4.2.1. *Gaussian Classifiers* 142
4.2.2. *The Curse of Dimensionality* 144
4.2.3. *Estimating the Error Rate* 146
4.2.4. *Comparing Classifiers* 148
4.3. DISCRIMINATIVE TRAINING 150
4.3.1. *Maximum Mutual Information Estimation* 150
4.3.2. *Minimum-Error-Rate Estimation* 156
4.3.3. *Neural Networks* 158
4.4. UNSUPERVISED ESTIMATION METHODS 163
4.4.1. *Vector Quantization* 163
4.4.2. *The EM Algorithm* 170
4.4.3. *Multivariate Gaussian Mixture Density Estimation* 172
4.5. CLASSIFICATION AND REGRESSION TREES 175
4.5.1. *Choice of Question Set* 177
4.5.2. *Splitting Criteria* 178
4.5.3. *Growing the Tree* 181
4.5.4. *Missing Values and Conflict Resolution* 182
4.5.5. *Complex Questions* 182
4.5.6. *The Right-Sized Tree* 184
4.6. HISTORICAL PERSPECTIVE AND FURTHER READING 190

PART II: SPEECH PROCESSING

5. DIGITAL SIGNAL PROCESSING 201

5.1. DIGITAL SIGNALS AND SYSTEMS 202
5.1.1. *Sinusoidal Signals* 203
5.1.2. *Other Digital Signals* 206
5.1.3. *Digital Systems* 206
5.2. CONTINUOUS-FREQUENCY TRANSFORMS 208
5.2.1. *The Fourier Transform* 208
5.2.2. *Z-Transform* 211
5.2.3. *Z-Transforms of Elementary Functions* 212
5.2.4. *Properties of the Z- and Fourier Transforms* 215
5.3. DISCRETE-FREQUENCY TRANSFORMS 216
5.3.1. *The Discrete Fourier Transform (DFT)* 218
5.3.2. *Fourier Transforms of Periodic Signals* 219
5.3.3. *The Fast Fourier Transform (FFT)* 222
5.3.4. *Circular Convolution* 227
5.3.5. *The Discrete Cosine Transform (DCT)* 228

5.4. DIGITAL FILTERS AND WINDOWS 229
5.4.1. *The Ideal Low-Pass Filter* 229
5.4.2. *Window Functions* 230
5.4.3. *FIR Filters* 232
5.4.4. *IIR Filters* 238
5.5. DIGITAL PROCESSING OF ANALOG SIGNALS 242
5.5.1. *Fourier Transform of Analog Signals* 243
5.5.2. *The Sampling Theorem* 243
5.5.3. *Analog-to-Digital Conversion* 245
5.5.4. *Digital-to-Analog Conversion* 246
5.6. MULTIRATE SIGNAL PROCESSING 248
5.6.1. *Decimation* 248
5.6.2. *Interpolation* 249
5.6.3. *Resampling* 250
5.7. FILTERBANKS 251
5.7.1. *Two-Band Conjugate Quadrature Filters* 251
5.7.2. *Multiresolution Filterbanks* 254
5.7.3. *The DFT as a Filterbank* 255
5.7.4. *Modulated Lapped Transforms* 258
5.8. STOCHASTIC PROCESSES 260
5.8.1. *Statistics of Stochastic Processes* 261
5.8.2. *Stationary Processes* 264
5.8.3. *LTI Systems with Stochastic Inputs* 267
5.8.4. *Power Spectral Density* 268
5.8.5. *Noise* 269
5.9. HISTORICAL PERSPECTIVE AND FURTHER READING 270

6. SPEECH SIGNAL REPRESENTATIONS 275

6.1. SHORT-TIME FOURIER ANALYSIS 276
6.1.1. *Spectrograms* 276
6.1.2. *Pitch-Synchronous Analysis* 281
6.2. ACOUSTICAL MODEL OF SPEECH PRODUCTION 283
6.2.1. *Glottal Excitation* 283
6.2.2. *Lossless Tube Concatenation* 284
6.2.3. *Source-Filter Models of Speech Production* 284
6.3. LINEAR PREDICTIVE CODING 288
6.3.1. *The Orthogonality Principle* 290
6.3.2. *Solution of the LPC Equations* 291
6.3.3. *Spectral Analysis via LPC* 292
6.3.4. *The Prediction Error* 300
6.3.5. *Equivalent Representations* 301
303

6.4. CEPSTRAL PROCESSING 306
6.4.1. *The Real and Complex Cepstrum* 307
6.4.2. *Cepstrum of Pole-Zero Filters* 308
6.4.3. *Cepstrum of Periodic Signals* 311
6.4.4. *Cepstrum of Speech Signals* 312
6.4.5. *Source-Filter Separation via the Cepstrum* 314
6.5. PERCEPTUALLY MOTIVATED REPRESENTATIONS 315
6.5.1. *The Bilinear Transform* 315
6.5.2. *Mel-Frequency Cepstrum* 316
6.5.3. *Perceptual Linear Prediction (PLP)* 318
6.6. FORMANT FREQUENCIES 319
6.6.1. *Statistical Formant Tracking* 320
6.7. THE ROLE OF PITCH 324
6.7.1. *Autocorrelation Method* 324
6.7.2. *Normalized Cross-Correlation Method* 327
6.7.3. *Signal Conditioning* 329
6.7.4. *Pitch Tracking* 330
6.8. HISTORICAL PERSPECTIVE AND FURTHER READING 332

7. SPEECH CODING 337

7.1. SPEECH CODERS ATTRIBUTES 338
7.2. SCALAR WAVEFORM CODERS 340
7.2.1. *Linear Pulse Code Modulation (PCM)* 340
7.2.2. *μ-law and A-law PCM* 342
7.2.3. *Adaptive PCM* 344
7.2.4. *Differential Quantization* 345
7.3. SCALAR FREQUENCY DOMAIN CODERS 348
7.3.1. *Benefits of Masking* 349
7.3.2. *Transform Coders* 350
7.3.3. *Consumer Audio* 351
7.3.4. *Digital Audio Broadcasting (DAB)* 352
7.4. CODE EXCITED LINEAR PREDICTION (CELP) 353
7.4.1. *LPC Vocoder* 353
7.4.2. *Analysis by Synthesis* 353
7.4.3. *Pitch Prediction: Adaptive Codebook* 356
7.4.4. *Perceptual Weighting and Postfiltering* 357
7.4.5. *Parameter Quantization* 358
7.4.6. *CELP Standards* 359
7.5. LOW-BIT RATE SPEECH CODERS 361
7.5.1. *Mixed-Excitation LPC Vocoder* 362
7.5.2. *Harmonic Coding* 363
7.5.3. *Waveform Interpolation* 367
7.6. HISTORICAL PERSPECTIVE AND FURTHER READING 371

PART III: SPEECH RECOGNITION

8. HIDDEN MARKOV MODELS 377

8.1. THE MARKOV CHAIN 378
8.2. DEFINITION OF THE HIDDEN MARKOV MODEL 380
8.2.1. Dynamic Programming and DTW 383
8.2.2. How to Evaluate an HMM—The Forward Algorithm 385
8.2.3. How to Decode an HMM—The Viterbi Algorithm 387
8.2.4. How to Estimate HMM Parameters—Baum-Welch Algorithm 389
8.3. CONTINUOUS AND SEMICONTINUOUS HMMS 394
8.3.1. Continuous Mixture Density HMMs 394
8.3.2. Semicontinuous HMMs 396
8.4. PRACTICAL ISSUES IN USING HMMS 398
8.4.1. Initial Estimates 398
8.4.2. Model Topology 399
8.4.3. Training Criteria 401
8.4.4. Deleted Interpolation 401
8.4.5. Parameter Smoothing 403
8.4.6. Probability Representations 404
8.5. HMM LIMITATIONS 405
8.5.1. Duration Modeling 406
8.5.2. First-Order Assumption 408
8.5.3. Conditional Independence Assumption 409
8.6. HISTORICAL PERSPECTIVE AND FURTHER READING 409

9. ACOUSTIC MODELING 415

9.1. VARIABILITY IN THE SPEECH SIGNAL 416
9.1.1. Context Variability 417
9.1.2. Style Variability 418
9.1.3. Speaker Variability 418
9.1.4. Environment Variability 419
9.2. HOW TO MEASURE SPEECH RECOGNITION ERRORS 419
9.3. SIGNAL PROCESSING—EXTRACTING FEATURES 421
9.3.1. Signal Acquisition 422
9.3.2. End-Point Detection 422
9.3.3. MFCC and Its Dynamic Features 424
9.3.4. Feature Transformation 426
9.4. PHONETIC MODELING—SELECTING APPROPRIATE UNITS 428
9.4.1. Comparison of Different Units 429
9.4.2. Context Dependency 430
9.4.3. Clustered Acoustic-Phonetic Units 432
9.4.4. Lexical Baseforms 436

9.5. ACOUSTIC MODELING—SCORING ACOUSTIC FEATURES 439
9.5.1. Choice of HMM Output Distributions 439
9.5.2. Isolated vs. Continuous Speech Training 441
9.6. ADAPTIVE TECHNIQUES—MINIMIZING MISMATCHES 444
9.6.1. Maximum a Posteriori (MAP) 445
9.6.2. Maximum Likelihood Linear Regression (MLLR) 447
9.6.3. MLLR and MAP Comparison 450
9.6.4. Clustered Models 452
9.7. CONFIDENCE MEASURES: MEASURING THE RELIABILITY 453
9.7.1. Filler Models 453
9.7.2. Transformation Models 454
9.7.3. Combination Models 456
9.8. OTHER TECHNIQUES 457
9.8.1. Neural Networks 457
9.8.2. Segment Models 459
9.9. CASE STUDY: WHISPER 464
9.10. HISTORICAL PERSPECTIVE AND FURTHER READING 465

10. ENVIRONMENTAL ROBUSTNESS 477

10.1. THE ACOUSTICAL ENVIRONMENT 478
10.1.1. Additive Noise 478
10.1.2. Reverberation 480
10.1.3. A Model of the Environment 482
10.2. ACOUSTICAL TRANSDUCERS 486
10.2.1. The Condenser Microphone 486
10.2.2. Directionality Patterns 489
10.2.3. Other Transduction Categories 496
10.3. ADAPTIVE ECHO CANCELLATION (AEC) 497
10.3.1. The LMS Algorithm 499
10.3.2. Convergence Properties of the LMS Algorithm 500
10.3.3. Normalized LMS Algorithm 501
10.3.4. Transform-Domain LMS Algorithm 502
10.3.5. The RLS Algorithm 503
10.4. MULTIMICROPHONE SPEECH ENHANCEMENT 504
10.4.1. Microphone Arrays 505
10.4.2. Blind Source Separation 510
10.5. ENVIRONMENT COMPENSATION PREPROCESSING 515
10.5.1. Spectral Subtraction 516
10.5.2. Frequency-Domain MMSE from Stereo Data 519
10.5.3. Wiener Filtering 520
10.5.4. Cepstral Mean Normalization (CMN) 522
10.5.5. Real-Time Cepstral Normalization 525
10.5.6. The Use of Gaussian Mixture Models 525

10.6. ENVIRONMENTAL MODEL ADAPTATION 528
10.6.1. *Retraining on Corrupted Speech* 528
10.6.2. *Model Adaptation* 530
10.6.3. *Parallel Model Combination* 531
10.6.4. *Vector Taylor Series* 535
10.6.5. *Retraining on Compensated Features* 537
10.7. MODELING NONSTATIONARY NOISE 538
10.8. HISTORICAL PERSPECTIVE AND FURTHER READING 540

11. LANGUAGE MODELING 545
11.1. FORMAL LANGUAGE THEORY 546
11.1.1. *Chomsky Hierarchy* 547
11.1.2. *Chart Parsing for Context-Free Grammars* 549
11.2. STOCHASTIC LANGUAGE MODELS 554
11.2.1. *Probabilistic Context-Free Grammars* 554
11.2.2. *N-gram Language Models* 558
11.3. COMPLEXITY MEASURE OF LANGUAGE MODELS 560
11.4. N-GRAM SMOOTHING 562
11.4.1. *Deleted Interpolation Smoothing* 564
11.4.2. *Backoff Smoothing* 565
11.4.3. *Class N-grams* 570
11.4.4. *Performance of N-gram Smoothing* 573
11.5. ADAPTIVE LANGUAGE MODELS 574
11.5.1. *Cache Language Models* 574
11.5.2. *Topic-Adaptive Models* 575
11.5.3. *Maximum Entropy Models* 576
11.6. PRACTICAL ISSUES 578
11.6.1. *Vocabulary Selection* 578
11.6.2. *N-gram Pruning* 580
11.6.3. *CFG vs. N-gram Models* 581
11.7. HISTORICAL PERSPECTIVE AND FURTHER READING 584

12. BASIC SEARCH ALGORITHMS 591
12.1. BASIC SEARCH ALGORITHMS 592
12.1.1. *General Graph Searching Procedures* 593
12.1.2. *Blind Graph Search Algorithms* 597
12.1.3. *Heuristic Graph Search* 601
12.2. SEARCH ALGORITHMS FOR SPEECH RECOGNITION 608
12.2.1. *Decoder Basics* 609
12.2.2. *Combining Acoustic and Language Models* 610
12.2.3. *Isolated Word Recognition* 610
12.2.4. *Continuous Speech Recognition* 611

12.3. LANGUAGE MODEL STATES 613
12.3.1. Search Space with FSM and CFG 613
12.3.2. Search Space with the Unigram 616
12.3.3. Search Space with Bigrams 617
12.3.4. Search Space with Trigrams 619
12.3.5. How to Handle Silences Between Words 621
12.4. TIME-SYNCHRONOUS VITERBI BEAM SEARCH 622
12.4.1. The Use of Beam 624
12.4.2. Viterbi Beam Search 625
12.5. STACK DECODING (A* SEARCH) 626
12.5.1. Admissible Heuristics for Remaining Path 630
12.5.2. When to Extend New Words 631
12.5.3. Fast Match 634
12.5.4. Stack Pruning 638
12.5.5. Multistack Search 639
12.6. HISTORICAL PERSPECTIVE AND FURTHER READING 640

13. LARGE-VOCABULARY SEARCH ALGORITHMS 645
13.1. EFFICIENT MANIPULATION OF A TREE LEXICON 646
13.1.1. Lexical Tree 646
13.1.2. Multiple Copies of Pronunciation Trees 648
13.1.3. Factored Language Probabilities 650
13.1.4. Optimization of Lexical Trees 653
13.1.5. Exploiting Subtree Polymorphism 656
13.1.6. Context-Dependent Units and Inter-Word Triphones 658
13.2. OTHER EFFICIENT SEARCH TECHNIQUES 659
13.2.1. Using Entire HMM as a State in Search 659
13.2.2. Different Layers of Beams 660
13.2.3. Fast Match 661
13.3. N-BEST AND MULTIPASS SEARCH STRATEGIES 663
13.3.1. N-best Lists and Word Lattices 664
13.3.2. The Exact N-best Algorithm 666
13.3.3. Word-Dependent N-best and Word-Lattice Algorithm 667
13.3.4. The Forward-Backward Search Algorithm 670
13.3.5. One-Pass vs. Multipass Search 673
13.4. SEARCH-ALGORITHM EVALUATION 674
13.5. CASE STUDY—MICROSOFT WHISPER 676
13.5.1. The CFG Search Architecture 676
13.5.2. The N-gram Search Architecture 677
13.6. HISTORICAL PERSPECTIVE AND FURTHER READING 681

PART IV: TEXT-TO-SPEECH SYSTEMS

14. TEXT AND PHONETIC ANALYSIS 689

14.1. MODULES AND DATA FLOW 690
14.1.1. Modules 692
14.1.2. Data Flows 694
14.1.3. Localization Issues 696
14.2. LEXICON 697
14.3. DOCUMENT STRUCTURE DETECTION 699
14.3.1. Chapter and Section Headers 700
14.3.2. Lists 701
14.3.3. Paragraphs 702
14.3.4. Sentences 702
14.3.5. Email 704
14.3.6. Web Pages 705
14.3.7. Dialog Turns and Speech Acts 705
14.4. TEXT NORMALIZATION 706
14.4.1. Abbreviations and Acronyms 709
14.4.2. Number Formats 712
14.4.3. Domain-Specific Tags 718
14.4.4. Miscellaneous Formats 719
14.5. LINGUISTIC ANALYSIS 720
14.6. HOMOGRAPH DISAMBIGUATION 724
14.7. MORPHOLOGICAL ANALYSIS 725
14.8. LETTER-TO-SOUND CONVERSION 728
14.9. EVALUATION 730
14.10. CASE STUDY: FESTIVAL 732
14.10.1. Lexicon 733
14.10.2. Text Analysis 733
14.10.3. Phonetic Analysis 735
14.11. HISTORICAL PERSPECTIVE AND FURTHER READING 735

15. PROSODY 739

15.1. THE ROLE OF UNDERSTANDING 740
15.2. PROSODY GENERATION SCHEMATIC 743
15.3. SPEAKING STYLE 744
15.3.1. Character 744
15.3.2. Emotion 744
15.4. SYMBOLIC PROSODY 744
15.4.1. Pauses 745
15.4.2. Prosodic Phrases 747
749

15.4.3. Accent 751
15.4.4. Tone 753
15.4.5. Tune 757
15.4.6. Prosodic Transcription Systems 759
15.5. DURATION ASSIGNMENT 761
15.5.1. Rule-Based Methods 762
15.5.2. CART-Based Durations 763
15.6. PITCH GENERATION 763
15.6.1. Attributes of Pitch Contours 764
15.6.2. Baseline F0 Contour Generation 768
15.6.3. Parametric F0 Generation 774
15.6.4. Corpus-Based F0 Generation 778
15.7. PROSODY MARKUP LANGUAGES 783
15.8. PROSODY EVALUATION 784
15.9. HISTORICAL PERSPECTIVE AND FURTHER READING 785

16. SPEECH SYNTHESIS 793

16.1. ATTRIBUTES OF SPEECH SYNTHESIS 794
16.2. FORMANT SPEECH SYNTHESIS 796
16.2.1. Waveform Generation from Formant Values 797
16.2.2. Formant Generation by Rule 800
16.2.3. Data-Driven Formant Generation 803
16.2.4. Articulatory Synthesis 803
16.3. CONCATENATIVE SPEECH SYNTHESIS 804
16.3.1. Choice of Unit 805
16.3.2. Optimal Unit String: The Decoding Process 810
16.3.3. Unit Inventory Design 817
16.4. PROSODIC MODIFICATION OF SPEECH 818
16.4.1. Synchronous Overlap and Add (SOLA) 818
16.4.2. Pitch Synchronous Overlap and Add (PSOLA) 820
16.4.3. Spectral Behavior of PSOLA 822
16.4.4. Synthesis Epoch Calculation 823
16.4.5. Pitch-Scale Modification Epoch Calculation 825
16.4.6. Time-Scale Modification Epoch Calculation 826
16.4.7. Pitch-Scale Time-Scale Epoch Calculation 827
16.4.8. Waveform Mapping 827
16.4.9. Epoch Detection 828
16.4.10. Problems with PSOLA 829
16.5. SOURCE-FILTER MODELS FOR PROSODY MODIFICATION 831
16.5.1. Prosody Modification of the LPC Residual 832
16.5.2. Mixed Excitation Models 832
16.5.3. Voice Effects 834

16.6. EVALUATION OF TTS SYSTEMS 834
16.6.1. *Intelligibility Tests* 837
16.6.2. *Overall Quality Tests* 840
16.6.3. *Preference Tests* 842
16.6.4. *Functional Tests* 842
16.6.5. *Automated Tests* 843
16.7. HISTORICAL PERSPECTIVE AND FURTHER READING 844

PART V: SPOKEN LANGUAGE SYSTEMS

17. SPOKEN LANGUAGE UNDERSTANDING 853

17.1. WRITTEN VS. SPOKEN LANGUAGES 855
17.1.1. *Style* 856
17.1.2. *Disfluency* 857
17.1.3. *Communicative Prosody* 858
17.2. DIALOG STRUCTURE 859
17.2.1. *Units of Dialog* 860
17.2.2. *Dialog (Speech) Acts* 861
17.2.3. *Dialog Control* 866
17.3. SEMANTIC REPRESENTATION 867
17.3.1. *Semantic Frames* 867
17.3.2. *Conceptual Graphs* 872
17.4. SENTENCE INTERPRETATION 873
17.4.1. *Robust Parsing* 874
17.4.2. *Statistical Pattern Matching* 878
17.5. DISCOURSE ANALYSIS 881
17.5.1. *Resolution of Relative Expression* 882
17.5.2. *Automatic Inference and Inconsistency Detection* 885
17.6. DIALOG MANAGEMENT 886
17.6.1. *Dialog Grammars* 887
17.6.2. *Plan-Based Systems* 888
17.6.3. *Dialog Behavior* 892
17.7. RESPONSE GENERATION AND RENDITION 894
17.7.1. *Response Content Generation* 895
17.7.2. *Concept-to-Speech Rendition* 899
17.7.3. *Other Renditions* 901
17.8. EVALUATION 901
17.8.1. *Evaluation in the ATIS Task* 901
17.8.2. *PARADISE Framework* 903
17.9. CASE STUDY—DR. WHO 906
17.9.1. *Semantic Representation* 906
17.9.2. *Semantic Parser (Sentence Interpretation)* 908

17.9.3. Discourse Analysis 909
17.9.4. Dialog Manager 910
17.10. HISTORICAL PERSPECTIVE AND FURTHER READING 913

18. APPLICATIONS AND USER INTERFACES 919

18.1. APPLICATION ARCHITECTURE 920
18.2. TYPICAL APPLICATIONS 921
18.2.1. Computer Command and Control 921
18.2.2. Telephony Applications 924
18.2.3. Dictation 926
18.2.4. Accessibility 929
18.2.5. Handheld Devices 930
18.2.6. Automobile Applications 930
18.2.7. Speaker Recognition 931
18.3. SPEECH INTERFACE DESIGN 931
18.3.1. General Principles 931
18.3.2. Handling Errors 937
18.3.3. Other Considerations 941
18.3.4. Dialog Flow 942
18.4. INTERNATIONALIZATION 943
18.5. CASE STUDY—MIPAD 945
18.5.1. Specifying the Application 946
18.5.2. Rapid Prototyping 948
18.5.3. Evaluation 949
18.5.4. Iterations 951
18.6. HISTORICAL PERSPECTIVE AND FURTHER READING 952

INDEX 957

Foreword

Recognition and understanding of spontaneous unrehearsed speech remains an elusive goal. To understand speech, a human considers not only the specific information conveyed to the ear, but also the context in which the information is being discussed. For this reason, people can understand spoken language even when the speech signal is corrupted by noise. However, understanding the context of speech is, in turn, based on a broad knowledge of the world. And this has been the source of the difficulty and over forty years of research.

It is difficult to develop computer programs that are sufficiently sophisticated to understand continuous speech by a random speaker. Only when programmers simplify the problem—by isolating words, limiting the vocabulary or number of speakers, or constraining the way in which sentences may be formed—is speech recognition by computer possible.

Since the early 1970s, researchers at AT&T, BBN, CMU, IBM, Lincoln Labs, MIT, and SRI have made major contributions in Spoken Language Understanding Research. In 1971, the Defense Advanced Research Projects Agency (DARPA) initiated an ambitious five-year, $15 million, multisite effort to develop speech understanding systems. The goals were to develop systems that would accept continuous speech from many speakers, with minimal speaker adaptation, and operate on a 1000-word vocabulary, artificial syntax, and a

constrained task domain. Two of the systems, Harpy and Hearsay-II, both developed at Carnegie Mellon University, achieved the original goals and in some instances surpassed them.

During the last three decades I have been at Carnegie Mellon, I have been very fortunate to be able to work with many brilliant students and researchers. Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon were arguably among the outstanding researchers in the speech group at CMU. Since then, they have moved to Microsoft and have put together a world-class team at Microsoft Research. Over the years, they have contributed standards for building spoken language understanding systems with Microsoft's SAPI/SDK family of products and pushed the technologies forward with the rest of the community. Today, they continue to play a premier leadership role in both the research community and in industry.

This new book, *Spoken Language Processing*, represents a welcome addition to the technical literature on this increasingly important emerging area of Information Technology. As we move from desktop PCs to personal digital assistants (PDAs), wearable computers, and Internet cell phones, speech becomes a central, if not the only, means of communication between the human and machine! Huang, Acero, and Hon have undertaken a commendable task of creating a comprehensive reference that covers theoretical, algorithmic, and systems aspects of the spoken language tasks of recognition, synthesis, and understanding.

The task of spoken language communication requires a system to recognize, interpret, execute, and respond to a spoken query. This task is complicated by the fact that the speech signal is corrupted by many sources: noise in the background, characteristics of the microphone, vocal tract characteristics of the speakers, and differences in pronunciation. In addition, the system has to cope with non-grammaticality of spoken communication and ambiguity of language. An effective system must strive to utilize all the available sources of knowledge—acoustics, phonetics and phonology, lexical, syntactic, and semantic structure of language, and task-specific context-dependent information.

Speech is based on a sequence of discrete sound segments that are linked in time. These segments, called phonemes, are assumed to have unique articulatory and acoustic characteristics. While the human vocal apparatus can produce an almost infinite number of articulatory gestures, the number of phonemes is limited. English as spoken in the United States, for example, contains 16 vowel and 24 consonant sounds. Each phoneme has distinguishable acoustic characteristics and, in combination with other phonemes, forms larger units such as syllables and words. Knowledge about the acoustic differences among these sound units is essential to distinguish one word from another, say, *bit* from *pit*.

When speech sounds are connected to form larger linguistic units, the acoustic characteristics of a given phoneme will change as a function of its immediate phonetic environment because of the interaction among various anatomical structures (such as the tongue, lips, and vocal chords) and their different degrees of sluggishness. The result is an overlap of phonemic information in the acoustic signal from one segment to the other. For example, the same underlying phoneme *t* can have drastically different acoustic characteristics in different words, say, in *tea*, *tree*, *city*, *beaten*, and *steep*. This effect, known as coarticulation, can occur within a given word or across a word boundary. Thus, the word *this* will have very different acoustic properties in phrases such as *this car* and *this ship*.

This book is self-contained for those who wish to familiarize themselves with the current state of spoken language systems technology. However, a researcher or a professional in the field will benefit from a thorough grounding in a number of disciplines, including:

- *Signal processing*: Fourier Transforms, DFT, and FFT
- *Acoustics*: physics of sounds and speech, models of vocal tract
- *Pattern recognition*: clustering and pattern matching techniques
- *Artificial intelligence*: knowledge representation and search, natural language processing
- *Computer science*: hardware, parallel systems, algorithm optimization
- *Statistics*: probability theory, hidden Markov models, dynamic programming
- *Linguistics*: acoustic phonetics, lexical representation, syntax, and semantics

A newcomer to this field, easily overwhelmed by the vast number of different algorithms scattered across many conference proceedings, can find in this book a set of techniques that Huang, Acero, and Hon have found to work well in practice. This book is unique in that it includes both the theory and implementation details necessary to build spoken language systems. If you were able to assemble all the individual material that is covered in the book and put it on a shelf, it would be several times larger than this volume and yet you would be missing vital information. You would not have the material that is in this book that threads it all into one story, one context. If you need additional resources, the authors include extensive references to get that additional detail. *Spoken Language Processing* is very appealing both as a textbook and as a reference book for practicing engineers. Some readers familiar with a specific topic may decide to skip a few chapters; others may want to focus in other chapters. This is not a book that you will pick up and read once from cover to cover, but one you will keep near you for reference as long as you work in this field.

Raj Reddy
Dean, School of Computer Science
Carnegie Mellon University

Preface

Our primary motivation in writing this book is to share our working experience to bridge the gap between the knowledge of industry gurus and newcomers to the spoken language processing community. Many powerful techniques hide in conference proceedings and academic papers for years before becoming widely recognized by the research community or the industry. We spent many years pursuing spoken language technology research at Carnegie Mellon University before we started spoken language R&D at Microsoft. We fully understand that it is by no means a small undertaking to transfer a state-of-the-art spoken language research system into a commercially viable product that can truly help people improve their productivity. Our experience in both industry and academia is reflected in the context of this book, which presents a contemporary and comprehensive description of both theoretic and practical issues in spoken language processing. This book is intended for people of diverse academic and practical backgrounds. Speech scientists, computer scientists, linguists, engineers, physicists, and psychologists all have a unique perspective on spoken language processing. This book will be useful to all of these special interest groups.

Spoken language processing is a diverse subject that relies on knowledge of many levels, including acoustics, phonology, phonetics, linguistics, semantics, pragmatics, and discourse. The diverse nature of spoken language processing requires knowledge in computer science, electrical engineering, mathematics, syntax, and psychology. There are a number of excellent books on the subfields of spoken language processing, including speech recognition, text-to-speech conversion, and spoken language understanding, but there is no single book that covers both theoretical and practical aspects of these subfields and spoken language interface design. We devote many chapters systematically introducing fundamental

theories needed to understand how speech recognition, text-to-speech synthesis, and spoken language understanding work. Even more important is the fact that the book highlights what works well in practice, which is invaluable if you want to build a practical speech recognizer, a practical text-to-speech synthesizer, or a practical spoken language system. Using numerous real examples in developing Microsoft's spoken language systems, we concentrate on showing how the fundamental theories can be applied to solve real problems in spoken language processing.

We would like to thank many people who helped us during our spoken language processing R&D careers. We are particularly indebted to Professor Raj Reddy at the School of Computer Science, Carnegie Mellon University. Under his leadership, Carnegie Mellon University has become a center of research excellence on spoken language processing. Today's computer industry and academia benefit tremendously from his leadership and contributions.

Special thanks are due to Microsoft for its encouragement of spoken language R&D. The management team at Microsoft has been extremely generous to the speech technology group. We are particularly grateful to Bill Gates, Nathan Myhrvold, Rick Rashid, Dan Ling, and Jack Breese for the great environment they have created for us at Microsoft Research. We would also like to thank Bob Muglia and Kai-Fu Lee for their leadership role in Microsoft's speech product development.

Scott Meredith helped us write a number of chapters in this book and deserves to be a co-author. His insight and experience in text-to-speech synthesis enriched this book a great deal. We also owe gratitude to many colleagues we worked with in the speech technology group of Microsoft Research. In alphabetic order, Jim Adcock, Bruno Alabiso, Fil Alleva, Eric Bidstrup, Antonio Bigazzi, Ciprian Chelba, Li Deng, James Droppo, Doug Duchene, Joshua Goodman, Mei-Yuh Hwang, Larry Israel, Derek Jacoby, Li Jiang, Yun-Cheng Ju, David Larson, Kevin Larson, Jingsong Liu, Ricky Loynd, Milind Mahajan, Peter Mau, John Merrill, Yunus Mohammed, Salman Mughal, Mike Plumpe, Scott Quinn, Bill Rockenbeck, Mike Rozak, Kevin Schofield, Roxana Teodorescu, Gina Venolia, Kuansan Wang, Ye-Yi Wang, and Shenzhi Zhang.

In addition, we want to thank Les Atlas, Jeff Bilmes, Alan Black, David Caulton, Eric Chang, Phil Chou, Dinei Florencio, Allen Gersho, Francisco Gimenez-Galanes, Hynek Hermansky, Henrique Malvar, Julian Odell, Mari Ostendorf, Joseph Pentheroudakis, Tandy Trower, and Charles Wayne. They provided us with many wonderful comments to refine this book. Tim Moore, Russ Hall, and Jane Bonnell at Prentice Hall helped us finish this book in a finite amount of time.

Finally, writing this book was a marathon that could not have been finished without the support of our spouses, Yingzhi, Donna, and Phen, during the many evenings and weekends we spent on this project.

Xuedong Huang
Alex Acero
Hsiao-Wuen Hon
Redmond, WA

CHAPTER 1

Introduction

From human prehistory to the new media of the future, speech communication has been and will be the dominant mode of human social bonding and information exchange. The spoken word is now extended, through technological mediation such as telephony, movies, radio, television, and the Internet. This trend reflects the primacy of spoken communication in human psychology.

In addition to human-human interaction, this human preference for spoken language communication finds a reflection in human-machine interaction as well. Most computers currently utilize a *graphical user interface* (GUI), based on graphically represented interface objects and functions such as windows, icons, menus, and pointers. Most computer operating systems and applications also depend on a user's keyboard strokes and mouse clicks, with a display monitor for feedback. Today's computers lack the fundamental human abilities to speak, listen, understand, and learn. Speech, supported by other natural modalities, will be one of the primary means of interfacing with computers. And, even before speech-based interaction reaches full maturity, applications in home, mobile, and office segments are incorporating spoken language technology to change the way we live and work.

A spoken language system needs to have both speech recognition and speech synthesis capabilities. However, those two components by themselves are not sufficient to build a useful spoken language system. An understanding and dialog component is required to manage interactions with the user; and domain knowledge must be provided to guide the system's interpretation of speech and allow it to determine the appropriate action. For all these components, significant challenges exist, including robustness, flexibility, ease of integration, and engineering efficiency. The goal of building commercially viable spoken language systems has long attracted the attention of scientists and engineers all over the world. The purpose of this book is to share our working experience in developing advanced spoken language processing systems with both our colleagues and newcomers. We devote many chapters to systematically introducing fundamental theories and to highlighting what works well based on numerous lessons we learned in developing Microsoft's spoken language systems.

1.1. MOTIVATIONS

What motivates the integration of spoken language as the primary interface modality? We present a number of scenarios, roughly in order of expected degree of technical challenges and expected time to full deployment.

1.1.1. Spoken Language Interface

There are generally two categories of users who can benefit from adoption of speech as a control modality in parallel with others, such as the mouse, keyboard, touch-screen, and joystick. For novice users, functions that are conceptually simple should be directly accessible. For example, raising the voice output volume under software control on the desktop speakers, a conceptually simple operation, in some GUI systems of today requires opening one or more windows or menus, and manipulating sliders, check-boxes, or other graphical elements. This requires some knowledge of the system's interface conventions and structures. For the novice user, to be able to say *raise the volume* would be more direct and natural. For expert users, the GUI paradigm is sometimes perceived as an obstacle or nuisance and shortcuts are sought. Frequently these shortcuts allow the power user's hands to remain on the keyboard or mouse while mixing content creation with system commands. For example, an operator of a graphic design system for CAD/CAM might wish to specify a text formatting command while keeping the pointer device in position over a selected screen element.

Speech has the potential to accomplish these functions more powerfully than keyboard and mouse clicks. Speech becomes more powerful when supplemented by information streams encoding other dynamic aspects of user and system status, which can be resolved by the semantic component of a complete multimodal interface. We expect such multimodal interactions to proceed based on more complete user modeling, including speech, visual orientation, natural and device-based gestures, and facial expression, and these will be coordinated with detailed system profiles of typical user tasks and activity patterns.

In some situations you must rely on speech as an input or output medium. For example, with wearable computers, it may be impossible to incorporate a large keyboard. When driving, safety is compromised by any visual distraction, and hands are required for controlling the vehicle. The ultimate speech-only device, the telephone, is far more widespread than the PC. Certain manual tasks may also require full visual attention to the focus of the work. Finally, spoken language interfaces offer obvious benefits for individuals challenged with a variety of physical disabilities, such as loss of sight or limitations in physical motion and motor skills. Chapter 18 contains a detailed discussion on spoken language applications.

1.1.2. Speech-to-Speech Translation

Speech-to-speech translation has been depicted for decades in science fiction stories. Imagine questioning a Chinese-speaking conversational partner by speaking English into an unobtrusive device, and hearing real-time replies you can understand. This scenario, like the spoken language interface, requires both speech recognition and speech synthesis technology. In addition, sophisticated multilingual spoken language understanding is needed. This highlights the need for tightly coupled advances in speech recognition, synthesis, and understanding systems, a point emphasized throughout this book.

1.1.3. Knowledge Partners

The ability of computers to process spoken language as proficient as humans will be a landmark to signal the arrival of truly intelligent machines. Alan Turing [29] introduced his famous *Turing test*. He suggested a game, in which a computer's use of language would form the criterion for intelligence. If the machine could win the game, it would be judged intelligent. In Turing's game, you play the role of an interrogator. By asking a series of questions via a teletype, you must determine the identity of the other two participants: a machine and a person. The task of the machine is to fool you into believing it is a person by responding as a person to your questions. The task of the other person is to convince you the other participant is the machine. The critical issue for Turing was that using language as humans do is sufficient as an operational test for intelligence.

The ultimate use of spoken language is to pass the Turing test in allowing future extremely intelligent systems to interact with human beings as knowledge partners in all aspects of life. This has been a staple of science fiction, but its day will come. Such systems require reasoning capabilities and extensive world knowledge embedded in sophisticated search, communication, and inference tools that are beyond the scope of this book. We expect that spoken language technologies described in this book will form the essential enabling mechanism to pass the Turing test.

1.2. SPOKEN LANGUAGE SYSTEM ARCHITECTURE

Spoken language processing refers to technologies related to speech recognition, text-to-speech, and spoken language understanding. A spoken language system has at least one of the following three subsystems: a speech recognition system that converts speech into words, a text-to-speech system that conveys spoken information, and a spoken language understanding system that maps words into actions and that plans system-initiated actions.

There is considerable overlap in the fundamental technologies for these three subareas. Manually created rules have been developed for spoken language systems with limited success. But, in recent decades, data-driven statistical approaches have achieved encouraging results, which are usually based on modeling the speech signal using well-defined statistical algorithms that can automatically extract knowledge from the data. The data-driven approach can be viewed fundamentally as a pattern recognition problem. In fact, speech recognition, text-to-speech conversion, and spoken language understanding can all be regarded as pattern recognition problems. The patterns are either recognized during the runtime operation of the system or identified during system construction to form the basis of runtime generative models such as prosodic templates needed for text-to-speech synthesis. While we use and advocate the statistical approach, we by no means exclude the knowledge engineering approach from consideration. If we have a good set of rules in a given problem area, there is no need to use the statistical approach at all. The problem is that, at time of this writing, we do not have enough knowledge to produce a complete set of high-quality rules. As scientific and theoretical generalizations are made from data collected to construct data-driven systems, better rules may be constructed. Therefore, the rule-based and statistical approaches are best viewed as complementary.

1.2.1. Automatic Speech Recognition

A source-channel mathematical model described in Chapter 3 is often used to formulate speech recognition problems. As illustrated in Figure 1.1, the speaker's mind decides the source word sequence $\mathbf{W}$ that is delivered through his/her text generator. The source is passed through a noisy communication channel that consists of the speaker's vocal apparatus to produce the speech waveform and the speech signal processing component of the speech recognizer. Finally, the speech decoder aims to decode the acoustic signal $\mathbf{X}$ into a word sequence $\hat{\mathbf{W}}$, which is hopefully close to the original word sequence $\mathbf{W}$.

A typical practical speech recognition system consists of basic components shown in the dotted box of Figure 1.2. Applications interface with the decoder to get recognition results that may be used to adapt other components in the system. *Acoustic models* include the representation of knowledge about acoustics, phonetics, microphone and environment variability, gender and dialect differences among speakers, etc. *Language models* refer to a system's knowledge of what constitutes a possible word, what words are likely to co-occur, and in what sequence. The semantics and functions related to an operation a user may wish to perform may also be necessary for the language model. Many uncertainties exist in these areas, associated with speaker characteristics, speech style and rate, recognition of basic

speech segments, possible words, likely words, unknown words, grammatical variation, noise interference, nonnative accents, and confidence scoring of results. A successful speech recognition system must contend with all of these uncertainties. But that is only the beginning. The acoustic uncertainties of the different accents and speaking styles of individual speakers are compounded by the lexical and grammatical complexity and variations of spoken language, which are all represented in the language model.

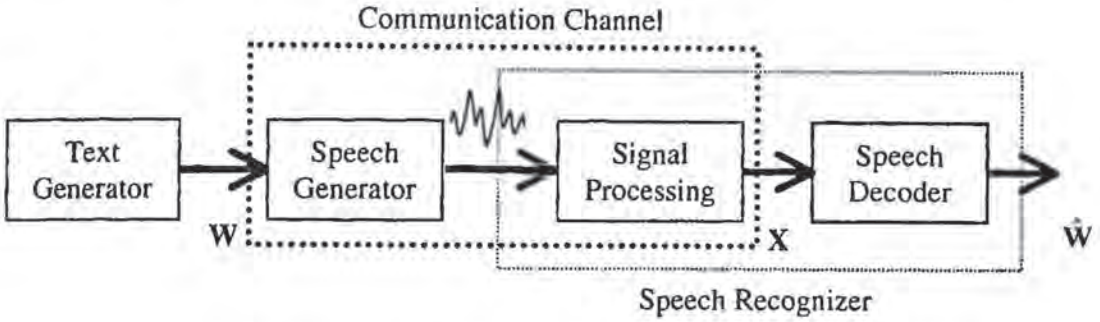


Figure 1.1 A source-channel model for a speech recognition system [15].

The speech signal is processed in the signal processing module that extracts salient feature vectors for the decoder. The decoder uses both acoustic and language models to generate the word sequence that has the maximum posterior probability for the input feature vectors. It can also provide information needed for the adaptation component to modify either the acoustic or language models so that improved performance can be obtained.

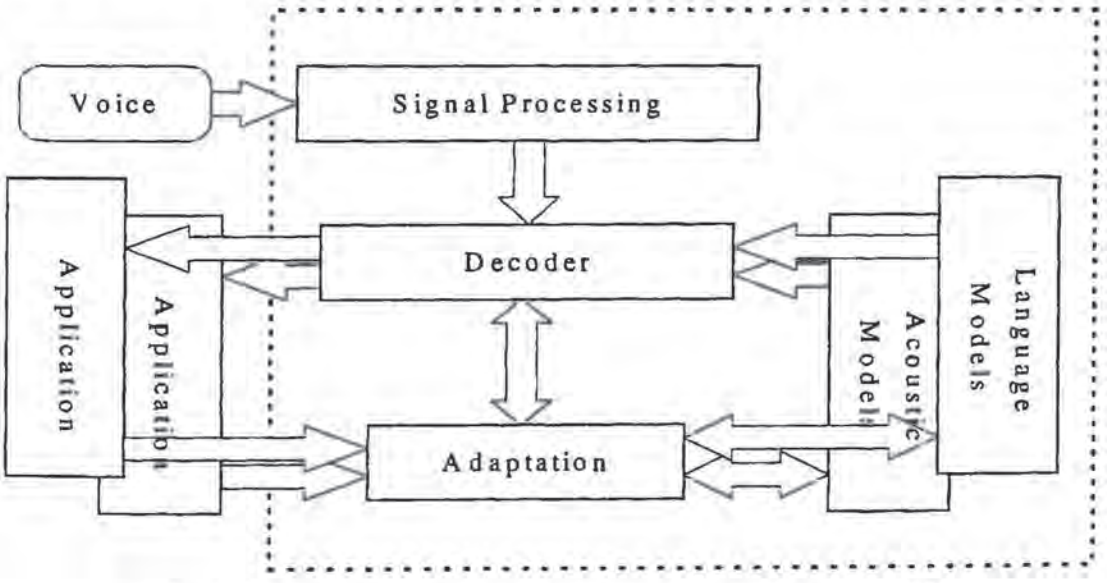


Figure 1.2 Basic system architecture of a speech recognition system [12].

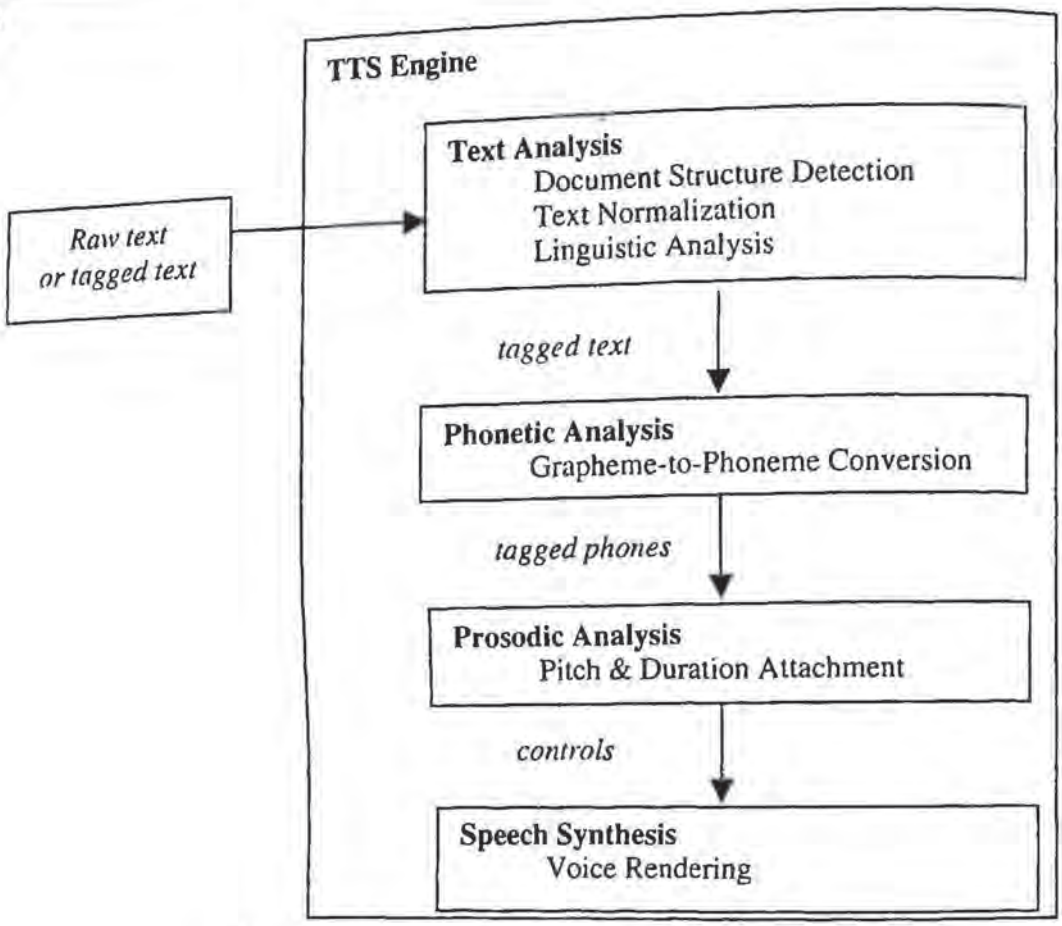


Figure 1.3 Basic system architecture of a TTS system.

1.2.2. Text-to-Speech Conversion

The term *text-to-speech*, often abbreviated as TTS, is easily understood. The task of a text-to-speech system can be viewed as speech recognition in reverse – a process of building a machinery system that can generate human-like speech from any text input to mimic human speakers. TTS is sometimes called *speech synthesis*, particularly in the engineering community.

The conversion of words in written form into speech is nontrivial. Even if we can store a huge dictionary for most common words in English; the TTS system still needs to deal with millions of names and acronyms. Moreover, in order to sound natural, the intonation of the sentences must be appropriately generated.

The development of TTS synthesis can be traced back to the 1930s when Dudley's *Voder*, developed by Bell Laboratories, was demonstrated at the World's Fair [18]. Taking advantage of increasing computation power and storage technology, TTS researchers have been able to generate high-quality commercial multilingual text-to-speech systems, although the quality is inferior to human speech for general-purpose applications.

The basic TTS components are shown in Figure 1.3. The text analysis component normalizes the text to the appropriate form so that it becomes speakable. The input can be

either raw text or tagged. These tags can be used to assist text, phonetic, and prosodic analysis. The phonetic analysis component converts the processed text into the corresponding phonetic sequence, which is followed by prosodic analysis to attach appropriate pitch and duration information to the phonetic sequence. Finally, the speech synthesis component takes the parameters from the fully tagged phonetic sequence to generate the corresponding speech waveform.

Various applications have different degrees of knowledge about the structure and content of the text that they wish to speak so some of the basic components shown in Figure 1.3 can be skipped. For example, some applications may have certain broad requirements such as rate and pitch. These requirements can be indicated with simple command tags appropriately located in the text. Many TTS systems provide a set of markups (tags), so the text producer can better express their semantic intention. An application may know a lot about the structure and content of the text to be spoken to greatly improve speech output quality. For engines providing such support, the *text analysis* phase can be skipped, in whole or in part. If the system developer knows the phonetic form, the phonetic analysis module can be skipped as well. The prosodic analysis module assigns a numeric duration to every phonetic symbol and calculates an appropriate pitch contour for the utterance or paragraph. In some cases, an application may have prosodic contours precalculated by some other process. This situation might arise when TTS is being used primarily for compression, or the prosody is *transplanted* from a real speaker's utterance. In these cases, the quantitative prosodic controls can be treated as special tagged field and sent directly along with the phonetic stream to speech synthesis for voice rendition.

1.2.3. Spoken Language Understanding

Whether a speaker is inquiring about flights to Seattle, reserving a table at a Pittsburgh restaurant, dictating an article in Chinese, or making a stock trade, a spoken language understanding system is needed to interpret utterances in context and carry out appropriate actions. Lexical, syntactic, and semantic knowledge must be applied in a manner that permits cooperative interaction among the various levels of acoustic, phonetic, linguistic, and application knowledge in minimizing uncertainty. Knowledge of the characteristic vocabulary, typical syntactic patterns, and possible actions in any given application context for both interpretation of user utterances and planning system activity are the heart and soul of any spoken language understanding system.

A schematic of a typical spoken language understanding system is shown in Figure 1.4. Such a system typically has a speech recognizer and a speech synthesizer for basic speech input and output, and a *sentence interpretation* component to parse the speech recognition results into semantic forms, which often need *discourse analysis* to track context and resolve ambiguities. The *Dialog Manager* is the central component that communicates with applications and the spoken language understanding modules such as discourse analysis, sentence interpretation, and response generation.

While most components of the system may be partly or wholly generic, the dialog manager controls the flow of conversation tied to the action. The dialog manager is respon-

sible for providing status needed for formulating responses, and maintaining the system's idea of the state of the discourse. The discourse state records the current transaction, dialog goals that motivated the current transaction, current objects in focus (temporary center of attention), the object history list for resolving dependent references, and other status information. The discourse information is crucial for sentence interpretation to interpret utterances in context. Various systems may alter the flow of information implied in Figure 1.4. For example, the dialog manager may be able to supply contextual discourse information or pragmatic inferences, as feedback to guide the recognizer's evaluation of hypotheses at the earliest level of search.

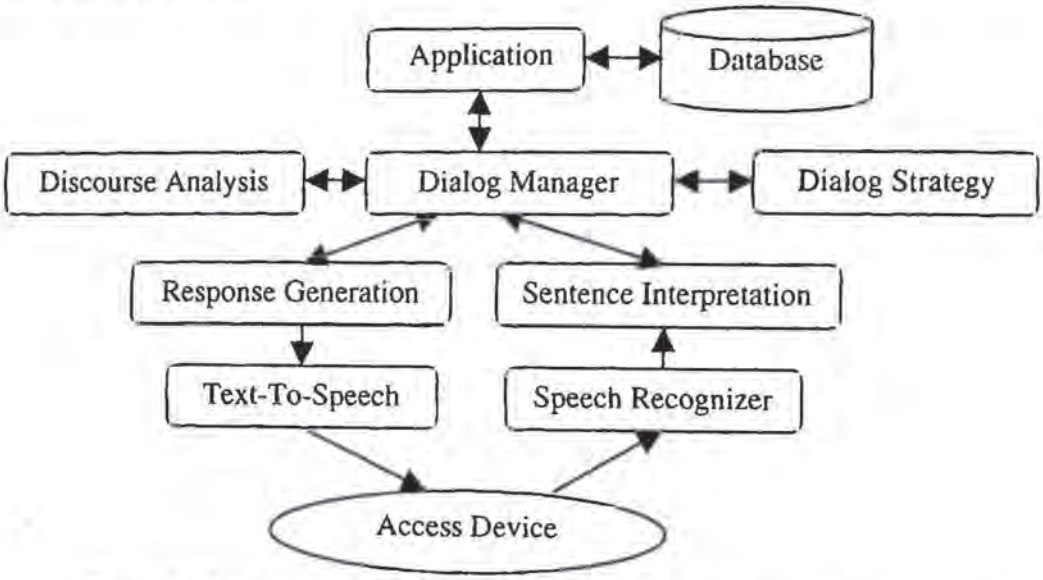


Figure 1.4 Basic system architecture of a spoken language understanding system.

1.3. BOOK ORGANIZATION

We attempt to present a comprehensive introduction to spoken language processing, which includes not only fundamentals but also a practical guide to build a working system that requires knowledge in speech signal processing, recognition, text-to-speech, spoken language understanding, and application integration. Since there is considerable overlap in the fundamental spoken language processing technologies, we have devoted Part I to the foundations needed. Part I contains background on speech production and perception, probability and information theory, and pattern recognition. Parts II, III, IV, and V include chapters on speech processing, speech recognition, speech synthesis, and spoken language systems, respectively. A reader with sufficient background can skip Part I, referring back to it later as needed. For example, the discussion of speech recognition in Part III relies on the pattern recognition algorithms presented in Part I. Algorithms that are used in several chapters

within Part III are also included in Parts I and II. Since the field is still evolving, at the end of each chapter we provide a historical perspective and list further readings to facilitate future research.

1.3.1. Part I: Fundamental Theory

Chapters 2 to 4 provide you with a basic theoretic foundation to better understand techniques that are widely used in modern spoken language systems. These theories include the essence of linguistics, phonetics, probability theory, information theory, and pattern recognition. These chapters prepare you fully to understand the rest of the book.

Chapter 2 discusses the basic structure of spoken language including speech science, phonetics, and linguistics. Chapter 3 covers probability theory and information theory, which form the foundation of modern pattern recognition. Many important algorithms and principles in pattern recognition and speech coding are derived based on these theories. Chapter 4 introduces basic pattern recognition, including decision theory, estimation theory, and a number of algorithms widely used in speech recognition. Pattern recognition forms the core of most of the algorithms used in spoken language processing.

1.3.2. Part II: Speech Processing

Part II provides you with necessary speech signal processing knowledge that is critical to spoken language processing. Most of what discuss here is traditionally the subject of electrical engineering.

Chapters 5 and 6 focus on how to extract useful information from the speech signal. The basic principles of digital signal processing are reviewed and a number of useful representations for the speech signal are discussed. Chapter 7 covers how to compress these representations for efficient transmission and storage.

1.3.3. Part III: Speech Recognition

Chapters 8 to 13 provide you with an in-depth look at modern speech recognition systems. We highlight techniques that have been proven to work well in real systems and explain in detail how and why these techniques work from both theoretic and practical perspectives.

Chapter 8 introduces hidden Markov models, the most prominent technique used in modern speech recognition systems. Chapters 9 and 11 deal with acoustic modeling and language modeling respectively. Because environment robustness is critical to the success of practical systems, we devote Chapter 10 to discussing how to make systems less affected by environment noises. Chapters 12 and 13 deal in detail with how to efficiently implement the decoder for speech recognition. Chapter 12 discusses a number of basic search algorithms, and Chapter 13 covers large vocabulary speech recognition. Throughout our discussion, Microsoft's Whisper speech recognizer is used as a case study to illustrate the methods introduced in these chapters.

1.3.4. Part IV: Text-to-Speech Systems

In Chapters 14 through 16, we discuss proven techniques in building text-to-speech systems. The synthesis system consists of major components found in speech recognition systems, except that they are in the reverse order.

Chapter 14 covers the analysis of written documents and the text needed to support spoken rendition, including the interpretation of audio markup commands, interpretation of numbers and other symbols, and conversion from orthographic to phonetic symbols. Chapter 15 focuses on the generation of pitch and duration controls for linguistic and emotional effect. Chapter 16 discusses the implementation of the synthetic voice, and presents algorithms to manipulate a limited voice data set to support a wide variety of pitch and duration controls required by the text analysis. We highlight the importance of trainable synthesis, with Microsoft's Whistler TTS system as an example.

1.3.5. Part V: Spoken Language Systems

As discussed in Section 1.1, spoken language applications motivate spoken language R&D. The central component is the spoken language understanding system. Since it is closely related to applications, we group it together with application and interface design.

Chapter 17 covers spoken language understanding. The output of the recognizer requires interpretation and action in a particular application context. This chapter details useful strategies for dialog management, and the coordination of all the speech and system resources to accomplish a task for a user. Chapter 18 concludes the book with a discussion of important principles for building spoken language interfaces and applications, including general human interface design goals, and interaction with other modalities in specific application contexts. Microsoft's MiPad is used as a case study to illustrate a number of issues in developing spoken language and multimodal applications.

1.4. TARGET AUDIENCES

This book can serve a variety of audiences:

Integration engineers: Software engineers who want to build spoken language systems, but who do not want to learn detailed speech technology internals, will find plentiful relevant material, including application design and software interfaces. Anyone with a professional interest in aspects of speech applications, integration, and interfaces can also achieve enough understanding of how the core technologies work, to allow them to take full advantage of state-of-the-art capabilities.

Speech technology engineers: Engineers and researchers working on various subspecialties within the speech field will find this book a useful guide to understanding related technologies in sufficient depth to help them gain insight on where their own approaches overlap with, or diverge from, their neighbors' common practice.

Graduate students: This book can serve as a primary textbook in a graduate or advanced undergraduate speech analysis or language engineering course. It can serve as a sup-

plementary textbook in some applied linguistics, digital signal processing, computer science, artificial intelligence, and possibly psycholinguistics course.

Linguists: As the practice of linguistics increasingly shifts to empirical analysis of real-world data, students and professional practitioners alike should find a comprehensive introduction to the technical foundations of computer processing of spoken language helpful. The book can be read at different levels and through different paths, for readers with differing technical skills and background knowledge.

Speech scientists: Researchers engaged in professional work on issues related to normal or pathological speech may find this complete exposition of the state-of-the-art in computer modeling of generation and perception of speech interesting.

Business planners: Increasingly, business and management functions require some level of insight into the vocabulary and common practices of technology development. While not the primary audience, managers, marketers, and others with planning responsibilities and sufficient technical background will find portions of this book useful in evaluating competing proposals, and in making business decisions related to the speech technology components.

1.5. HISTORICAL PERSPECTIVE AND FURTHER READING

Spoken language processing is a diverse field that relies on knowledge of language at the levels of signal processing, acoustics, phonology, phonetics, syntax, semantics, pragmatics, and discourse. The foundations of spoken language processing lie in computer science, electrical engineering, linguistics, and psychology. In the 1970s an ambitious speech understanding project was funded by DARPA, which led to many seminal systems and technologies [17]. A number of human language technology projects funded by DARPA in the 1980s and 1990s further accelerated the progress, as evidenced by many papers published in *The Proceedings of the DARPA Speech and Natural Language/Human Language Workshop*. The field is still rapidly progressing and there are a number of excellent review articles and introductory books. We provide a brief list here. More detailed references can be found within each chapter of this book. Gold and Morgan's *Speech and Audio Signal Processing* [10] also has a strong historical perspective on spoken language processing.

Hyde [14] and Reddy [24] provided an excellent review of early speech recognition work in the 1970s. Some of the principles are still applicable to today's speech recognition research. Waibel and Lee assembled many seminal papers in *Readings in Speech Recognition Speech Recognition* [31]. There are a number of excellent books on modern speech recognition [1, 13, 15, 22, 23].

Where does the state of the art speech recognition system stand today? A number of different recognition tasks can be used to compare the recognition error rate of people vs. machines. Table 1.1 shows five typical recognition tasks with vocabularies ranging from 10 to 5000 words speaker-independent continuous speech recognition. The Wall Street Journal Dictation (WSJ) Task has a 5000-word vocabulary as a continuous dictation application for the WSJ articles. In Table 1.1, the error rate for machines is based on state of the art speech

recognizers such as systems described in Chapter 9, and the error rate of humans is based on a range of subjects tested on the similar task. We can see the error rate of humans is at least 5 times smaller than machines except for the sentences that are generated from a trigram language model, where the sentences have the perfect match between humans and machines so humans cannot use high-level knowledge that is not used in machines.[1]

Table 1.1 Word error rate comparisons between human and machines on similar tasks.

Tasks	Vocabulary	Humans	Machines
Connected digits	10	0.009%	0.72%
Alphabet letters	26	1%	5%
Spontaneous telephone speech	2000	3.8%	36.7%
WSJ with clean speech	5000	0.9%	4.5%
WSJ with noisy speech (10-db SNR)	5000	1.1%	8.6%
Clean speech based on trigram sentences	20,000	7.6%	4.4%

We can see that humans are far more robust than machines for normal tasks. The error rate for machine spontaneous conversational telephone speech recognition is above 35%, more than a factor 10 higher than humans on the similar task. In addition, the error rate of humans does not increase as dramatically as machines when the environment becomes noisy (from quiet to 10-db SNR environments on the WSJ task). The relative error rate of humans increases from 0.9% to 1.1% (1.2 times), while the error rate of CSR systems increases from 4.5% to 8.6% (1.9 times). One interesting experiment is that when we generated sentences using the WSJ trigram language model (cf. Chapter 11), the difference between humans and machines disappears (the last row in Table 1.1). In fact, the error rate of humans is even higher than machines. This is because both humans and machines have the same high-level syntactic and semantic models. The test sentences are somewhat random to humans but perfect to machines that used the same trigram model for decoding. This experiment indicates humans make more effective use of semantic and syntactic constraints for improved speech recognition in meaningful conversation. In addition, machines don't have attention problems as humans do on random sentences.

Fant [7] gave an excellent introduction to speech production. Early reviews of text-to-speech synthesis can be found in [3, 8, 9]. Sagisaka [26] and Carlson [6] provide more recent reviews of progress in speech synthesis. A more detailed treatment can be found in [19, 30].

Where does the state of the art text to speech system stand today? Unfortunately, like speech recognition, this is not a solved problem either. Although machine storage capabilities are improving, the quality remains a challenge for many researchers if we want to pass the Turing test.

[1] Some of these experiments were conducted at Microsoft with only a small number of human subjects (3-5 people), which is not statistically significant. Nevertheless, the experiments give some interesting insight on the performance of humans and machines.

Spoken language understanding is deeply rooted in speech recognition research. There are a number of good books on spoken language understanding [2, 5, 16]. Manning and Schutze [20] focuses on statistical methods for language understanding. Like Waibel and Lee, Grosz et al. assembled many foundational papers in *Readings in Natural Language Processing* [11]. More recent reviews of progress in spoken language understanding can be found in [25, 28]. Related spoken language interface design issues can be found in [4, 21, 27, 32].

In comparison to speech recognition and text to speech, spoken language understanding is further away from approaching the level of humans, especially for general-purpose spoken language applications.

A number of good conference proceedings and journals report the latest progress in the field. Major results on spoken language processing are presented at the *International Conference on Acoustics, Speech and Signal Processing (ICASSP), International Conference on Spoken Language Processing (ICSLP), Eurospeech Conference,* the *DARPA Speech and Human Language Technology Workshops,* and many workshops organized by the *European Speech Communications Associations (ESCA)* and *IEEE Signal Processing Society.* Journals include *IEEE Transactions on Speech and Audio Processing, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Computer Speech and Language, Speech Communication,* and *Journal of Acoustical Society of America (JASA).* Research results can also be found at computational linguistics conferences such as the *Association for Computational Linguistics (ACL), International Conference on Computational Linguistics (COLING),* and *Applied Natural Language Processing (ANLP).* The journals *Computational Linguistics* and *Natural Language Engineering* cover both theoretical and practical applications of language research. *Speech Recognition Update* published by TMA Associates is an excellent industry newsletter on spoken language applications.

REFERENCES

[1] Acero, A., *Acoustical and Environmental Robustness in Automatic Speech Recognition*, 1993, Boston, MA, Kluwer Academic Publishers.

[2] Allen, J., *Natural Language Understanding*, 2nd ed., 1995, Menlo Park, CA, The Benjamin/Cummings Publishing Company.

[3] Allen, J., M.S. Hunnicutt, and D.H. Klatt, *From Text to Speech: The MITalk System*, 1987, Cambridge, UK, University Press.

[4] Balentine, B., and D. Morgan, *How to Build a Speech Recognition Application*, 1999, Enterprise Integration Group.

[5] Bernsen, N., H. Dybkjar, and L. Dybkjar, *Designing Interactive Speech Systems*, 1998, Springer.

[6] Carlson, R., "Models of Speech Synthesis" in *Voice Communications Between Humans and Machines. National Academy of Sciences*, D.B. Roe and J.G. Wilpon, eds., 1994, Washington, D.C., National Academy of Sciences.

[7] Fant, G., *Acoustic Theory of Speech Production*, 1970, The Hague, NL, Mouton.

[8] Flanagan, J., *Speech Analysis Synthesis and Perception*, 1972, New York, Springer-Verlag.

[9] Flanagan, J., "Voices Of Men And Machines," *Journal of Acoustical Society of America*, 1972, **51**, p. 1375.

[10] Gold, B. and N. Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2000, John Wiley and Sons.

[11] Grosz, B., F.S. Jones, and B.L. Webber, *Readings in Natural Language Processing*, 1986, Los Altos, CA, Morgan Kaufmann.

[12] Huang, X., *et al.*, "From Sphinx-II to Whisper -- Make Speech Recognition Usable" in *Automatic Speech and Speaker Recognition*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds. 1996, Norwell, MA, Kluwer Academic Publishers.

[13] Huang, X.D., Y. Ariki, and M.A. Jack, *Hidden Markov Models for Speech Recognition*, 1990, Edinburgh, U.K., Edinburgh University Press.

[14] Hyde, S.R., "Automatic Speech Recognition: Literature, Survey, and Discussion" in *Human Communication, A Unified Approach*, E.E. David and P.B. Denes, eds. 1972, New York, McGraw Hill.

[15] Jelinek, F., *Statistical Methods for Speech Recognition*, Language, Speech, and Communication, 1998, Cambridge, MA, MIT Press.

[16] Jurafsky, D. and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2000, Upper Saddle River, NJ, Prentice Hall.

[17] Klatt, D., "Review of the ARPA Speech Understanding Project," *Journal of Acoustical Society of America*, 1977, **62**(6), pp. 1324-1366.

[18] Klatt, D., "Review of Text-to-Speech Conversion for English," *Journal of Acoustical Society of America*, 1987, **82**, pp. 737-793.

[19] Kleijn, W.B. and K.K. Paliwal, *Speech Coding and Synthesis*, 1995, Amsterdam, Netherlands, Elsevier.

[20] Manning, C. and H. Schutze, *Foundations of Statistical Natural Language Processing*, 1999, MIT Press, Cambridge, USA.

[21] Markowitz, J., *Using Speech Recognition*, 1996, Prentice Hall.

[22] Mori, R.D., *Spoken Dialogues with Computers*, 1998, London, UK, Academic Press.

[23] Rabiner, L.R. and B.H. Juang, *Fundamentals of Speech Recognition*, May, 1993, Prentice-Hall.

[24] Reddy, D.R., "Speech Recognition by Machine: A Review," *IEEE Proc.*, 1976, **64**(4), pp. 502-531.

[25] Sadek, D. and R.D. Mori, "Dialogue Systems" in *Spoken Dialogues with Computers*, R.D. Mori, Editor 1998, London, UK, pp. 523-561, Academic Press.

[26] Sagisaka, Y., "Speech Synthesis from Text," *IEEE Communication Magazine*, 1990(1).

[27] Schmandt, C., *Voice Communication with Computers*, 1994, New York, NY, Van Nostrand Reinhold.

[28] Seneff, S., "The Use of Linguistic Hierarchies in Speech Understanding," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[29] Turing, A.M., "Computing Machinery and Intelligence," *Mind*, 1950, **LIX**(236), pp. 433-460.

[30] van Santen, J., *et al.*, *Progress in Speech Synthesis*, 1997, New York, Springer-Verlag.

[31] Waibel, A.H. and K.F. Lee, *Readings in Speech Recognition*, 1990, San Mateo, CA, Morgan Kaufman Publishers.

[32] Weinschenk, S. and D. Barker, *Designing Effective Speech Interfaces*, 2000, John Wiley & Sons, Inc.

PART I

FUNDAMENTAL THEORY

CHAPTER 2

Spoken Language Structure

Spoken language is used to communicate information from a speaker to a listener. Speech production and perception are both important components of the speech chain. Speech begins with a thought and intent to communicate in the brain, which activates muscular movements to produce speech sounds. A listener receives it in the auditory system, processing it for conversion to neurological signals the brain can understand. The speaker continuously monitors and controls the vocal organs by receiving his or her own speech as feedback.

Considering the universal components of speech communication as shown in Figure 2.1, the fabric of spoken interaction is woven from many distinct elements. The speech production process starts with the semantic message in a person's mind to be transmitted to the listener via speech. The computer counterpart to the process of message formulation is the application semantics that creates the concept to be expressed. After the message is created,

the next step is to convert the message into a sequence of words. Each word consists of a sequence of phonemes that corresponds to the pronunciation of the words. Each sentence also contains a prosodic pattern that denotes the duration of each phoneme, intonation of the sentence, and loudness of the sounds. Once the language system finishes the mapping, the talker executes a series of neuromuscular signals. The neuromuscular commands perform articulatory mapping to control the vocal cords, lips, jaw, tongue, and velum, thereby producing the sound sequence as the final output. The speech understanding process works in reverse order. First the signal is passed to the cochlea in the inner ear, which performs frequency analysis as a filter bank. A neural transduction process follows and converts the spectral signal into activity signals on the auditory nerve, corresponding roughly to a feature extraction component. Currently, it is unclear how neural activity is mapped into the language system and how message comprehension is achieved in the brain.

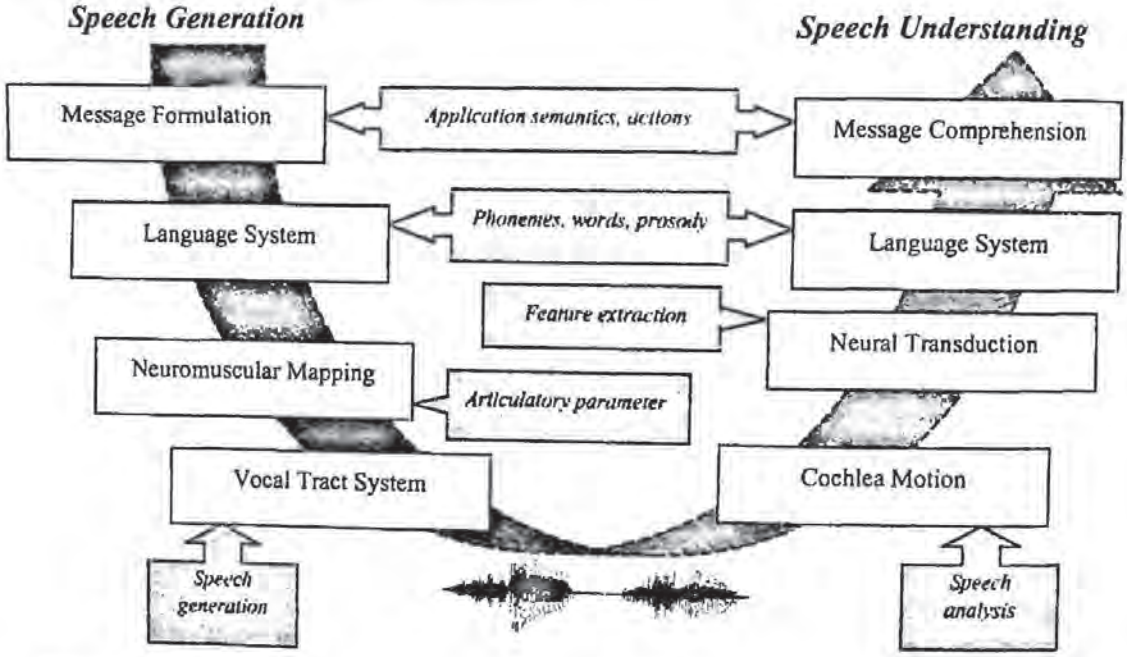


Figure 2.1 The underlying determinants of speech generation and understanding. The gray boxes indicate the corresponding computer system components for spoken language processing.

Speech signals are composed of analog sound patterns that serve as the basis for a discrete, symbolic representation of the spoken language – phonemes, syllables, and words. The production and interpretation of these sounds are governed by the syntax and semantics of the language spoken. In this chapter, we take a bottom up approach to introduce the basic concepts from sound to phonetics and phonology. Syllables and words are followed by syntax and semantics, which form the structure of spoken language processing. The examples in this book are drawn primarily from English, though they are relevant to other languages.

2.1. SOUND AND HUMAN SPEECH SYSTEMS

In this section, we briefly review human speech production and perception systems. We hope spoken language research will enable us to build a computer system that is as good as or better than our own speech production and understanding system.

2.1.1. Sound

Sound is a longitudinal pressure wave formed of compressions and rarefactions of air molecules, in a direction parallel to that of the application of energy. Compressions are zones where air molecules have been forced by the application of energy into a tighter-than-usual configuration, and rarefactions are zones where air molecules are less tightly packed. The alternating configurations of compression and rarefaction of air molecules along the path of an energy source are sometimes described by the graph of a sine wave as shown in Figure 2.2. In this representation, crests of the sine curve correspond to moments of maximal compression and troughs to moments of maximal rarefaction.

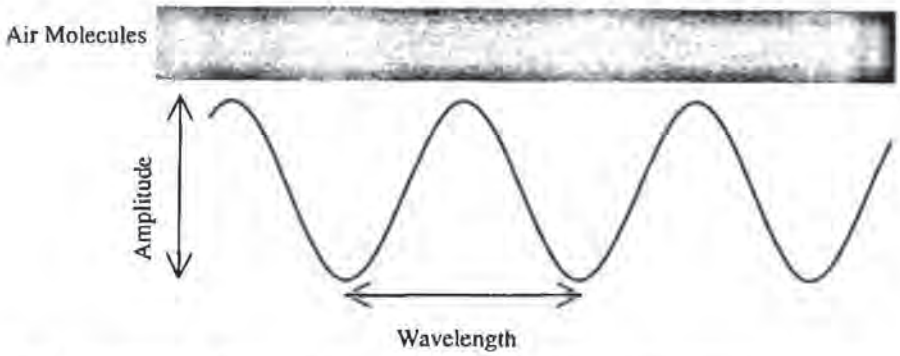


Figure 2.2 Application of sound energy causes alternating compression/rarefaction of air molecules, described by a sine wave. There are two important parameters, amplitude and wavelength, to describe a sine wave. Frequency [cycles/second measured in Hertz (Hz)] is also used to measure of the waveform.

The use of the sine graph in Figure 2.2 is only a notational convenience for charting local pressure variations over time, since sound does not form a transverse wave, and the air particles are just *oscillating in place* along the line of application of energy. The speed of a sound pressure wave in air is approximately $331.5 + 0.6T_c m/s$, where T_c is the Celsius temperature.

The amount of work done to generate the energy that sets the air molecules in motion is reflected in the amount of displacement of the molecules from their resting position. This *degree of displacement* is measured as the amplitude of a sound as shown in Figure 2.2. Because of the wide range, it is convenient to measure sound amplitude on a logarithmic scale in *decibels* (dB). A decibel scale is a means for comparing the intensity of two sounds:

$$10 \log_{10} (I / I_0) \tag{2.1}$$

where I and I_0 are the two intensity levels, with intensity being proportional to the square of the sound pressure P.

Sound pressure level (SPL) is a measure of absolute sound pressure P in dB:

$$SPL(dB) = 20 \log_{10} \left(\frac{P}{P_0} \right) \tag{2.2}$$

where the reference 0 dB corresponds to the threshold of hearing, which is $P_0 = 0.0002 \mu bar$ for a tone of 1kHz. The speech conversation level at 3 feet is about 60 dB SPL, and a jackhammer's level is about 120 dB SPL. Alternatively, watts/meter^2 units are often used to indicate intensity. We can bracket the limits of human hearing as shown in Table 2.1. On the low end, the human ear is quite sensitive. A typical person can detect sound waves having an intensity of 10^{-12} W/m^2 (the *threshold of hearing* or TOH). This intensity corresponds to a pressure wave affecting a given region by only one-billionth of a centimeter of molecular motion. On the other end, the most intense sound that can be safely detected without suffering physical damage is one trillion times more intense than the TOH. 0 dB begins with the TOH and advances logarithmically. The faintest audible sound is arbitrarily assigned a value of 0 dB, and the loudest sounds that the human ear can tolerate are about 120 dB.

Table 2.1 Intensity and decibel levels of various sounds.

Sound	dB Level	Times > TOH
Threshold of hearing (TOH: $10^{-12} W/m^2$)	0	10^0
Light whisper	10	10^1
Quiet living room	20	10^2
Quiet conversation	40	10^4
Average office	50	10^5
Normal conversation	60	10^6
Busy city street	70	10^7
Acoustic guitar – 1 ft. away	80	10^8
Heavy truck traffic	90	10^9
Subway from platform	100	10^{10}
Power tools	110	10^{11}
Pain threshold of ear	120	10^{12}
Airport runway	130	10^{13}
Sonic boom	140	10^{14}
Permanent damage to hearing	150	10^{15}
Jet engine, close up	160	10^{16}
Rocket engine	180	10^{18}
Twelve ft. from artillery cannon muzzle ($10^{10} W/m^2$)	220	10^{22}

The absolute threshold of hearing is the maximum amount of energy of a pure tone that cannot be detected by a listener in a noise free environment. The absolute threshold of hearing is a function of frequency that can be approximated by

$$T_q(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4 \quad (dB\ SPL) \tag{2.3}$$

and is plotted in Figure 2.3.

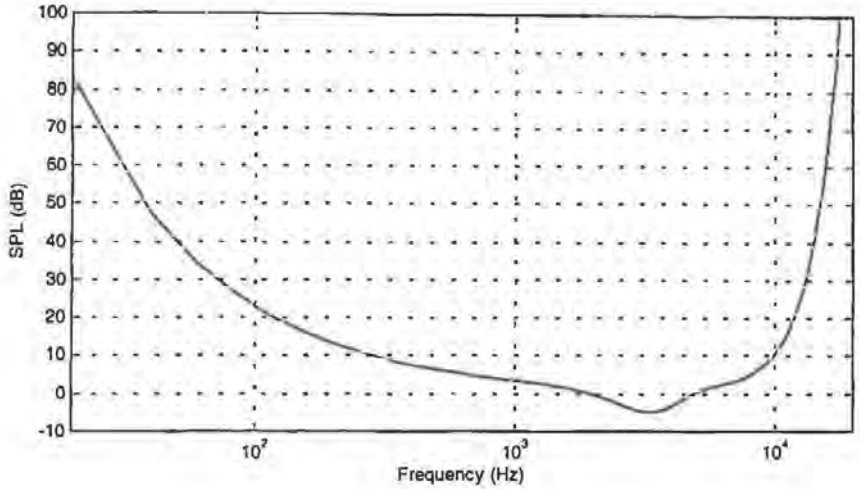


Figure 2.3 The sound pressure level (SPL) level in dB of the absolute threshold of hearing as a function of frequency. Sounds below this level are inaudible. Note that below 100 Hz and above 10 kHz this level rises very rapidly. Frequency goes from 20 Hz to 20 kHz and is plotted in a logarithmic scale from Eq. (2.3).

Let's compute how the pressure level varies with distance for a sound wave emitted by a point source located a distance r away. Assuming no energy absorption or reflection, the sound wave of a point source is propagated in a spherical front, such that the energy is the same for the sphere's surface at all radius r. Since the surface of a sphere of radius r is $4\pi r^2$, the sound's energy is inversely proportional to r^2, so that every time the distance is doubled, the sound pressure level decreases by 6 dB. For the point sound source, the energy (E) transported by a wave is proportional to the square of the amplitude (A) of the wave and the distance (r) between the sound source and the listener:

$$E \propto \frac{A^2}{r^2} \tag{2.4}$$

The typical sound intensity of a speech signal one inch away (close-talking microphone) from the talker is 1 Pascal = 10μbar, which corresponds to 94 dB SPL. The typical sound intensity 10 inches away from a talker is 0.1 Pascal = 1μbar, which corresponds to 74 dB SPL.

2.1.2. Speech Production

We review here basic human speech production systems, which have influenced research on speech coding, synthesis, and recognition.

2.1.2.1. Articulators

Speech is produced by air-pressure waves emanating from the mouth and the nostrils of a speaker. In most of the world's languages, the inventory of *phonemes,* as discussed in Section 2.2.1, can be split into two basic classes:

- consonants – articulated in the presence of constrictions in the throat or obstructions in the mouth (tongue, teeth, lips) as we speak.
- vowels – articulated without major constrictions and obstructions.

The sounds can be further partitioned into subgroups based on certain articulatory properties. These properties derive from the anatomy of a handful of important articulators and the places where they touch the boundaries of the human vocal tract. Additionally, a large number of muscles contribute to articulator positioning and motion. We restrict ourselves to a schematic view of only the major articulators, as diagrammed in Figure 2.4. The

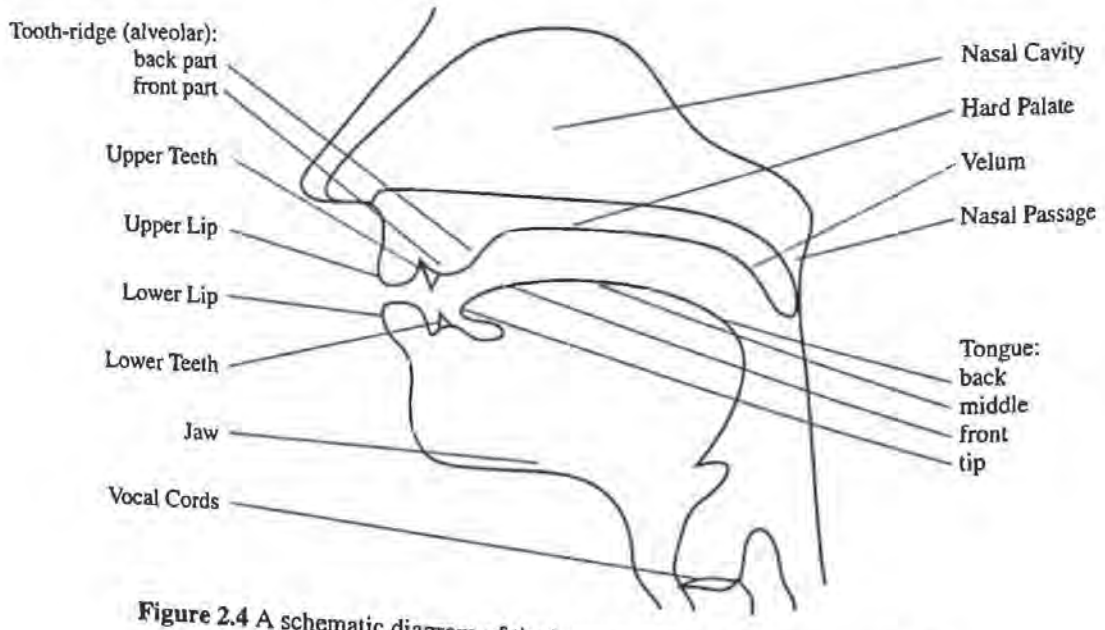


Figure 2.4 A schematic diagram of the human speech production apparatus.

gross components of the speech production apparatus are the lungs, trachea, larynx (organ of voice production), pharyngeal cavity (throat), oral and nasal cavity. The pharyngeal and oral cavities are typically referred to as the vocal tract, and the nasal cavity as the nasal tract. As illustrated in Figure 2.4, the human speech production apparatus consists of:

- *Lungs*: source of air during speech.
- *Vocal cords (larynx):* when the vocal folds are held close together and oscillate against one another during a speech sound, the sound is said to be *voiced.* When the folds are too slack or tense to vibrate periodically, the sound is said to be *unvoiced.* The place where the vocal folds come together is called the *glottis.*
- *Velum (soft palate)*: operates as a *valve*, opening to allow passage of air (and thus resonance) through the nasal cavity. Sounds produced with the flap open include *m* and *n*.
- *Hard palate*: a long relatively hard surface at the roof inside the mouth, which, when the tongue is placed against it, enables consonant articulation.
- *Tongue*: flexible articulator, shaped away from the palate for vowels, placed close to or on the palate or other hard surfaces for consonant articulation.
- *Teeth*: another place of articulation used to brace the tongue for certain consonants.
- *Lips*: can be rounded or spread to affect vowel quality, and closed completely to stop the oral air flow in certain consonants (*p, b, m*).

2.1.2.2. The Voicing Mechanism

The most fundamental distinction between sound types in speech is the voiced/voiceless distinction. Voiced sounds, including vowels, have in their time and frequency structure a roughly regular pattern that voiceless sounds, such as consonants like *s*, lack. Voiced sounds typically have more energy as shown in Figure 2.5. We see here the waveform of the word *sees*, which consists of three phonemes: an unvoiced consonant */s/*, a vowel */iy/*, and a voiced consonant */z/*.

What in the speech production mechanism creates this fundamental distinction? When the vocal folds vibrate during phoneme articulation, the phoneme is considered voiced; otherwise it is unvoiced. Vowels are voiced throughout their duration. The distinct vowel *timbres* are created by using the tongue and lips to shape the main oral resonance cavity in different ways. The vocal folds vibrate at slower or faster rates, from as low as 60 cycles per second (Hz) for a large man, to as high as 300 Hz or higher for a small woman or child. The rate of cycling (opening and closing) of the vocal folds in the larynx during phonation of voiced sounds is called the *fundamental frequency*. This is because it sets the periodic baseline for all higher-frequency harmonics contributed by the pharyngeal and oral resonance

cavities above. The fundamental frequency also contributes more than any other single factor to the perception of *pitch* (the semi-musical rising and falling of voice tones) in speech.

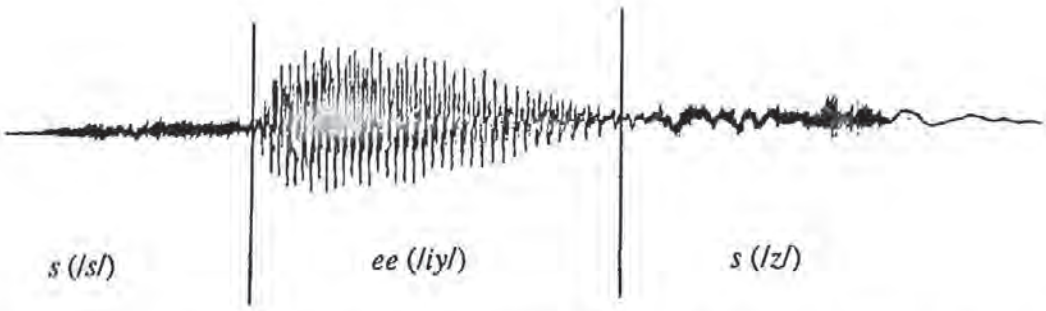


Figure 2.5 Waveform of *sees*, showing a voiceless phoneme /s/, followed by a voiced sound, the vowel /iy/. The final sound, /z/, is a type of voiced consonant.

The glottal cycle is illustrated in Figure 2.6. At stage (a), the vocal folds are closed and the air stream from the lungs is indicated by the arrow. At some point, the air pressure on the underside of the barrier formed by the vocal folds increases until it overcomes the resistance of the vocal fold closure and the higher air pressure below blows them apart (b). However, the tissues and muscles of the larynx and the vocal folds have a natural elasticity which tends to make them fall back into place rapidly, once air pressure is temporarily equalized (c). The successive airbursts resulting from this process are the source of energy for all voiced sounds. The time for a single open-close cycle depends on the stiffness and size of the vocal folds and the amount of subglottal air pressure. These factors can be controlled by a speaker to raise and lower the perceived frequency or pitch of a voiced sound.

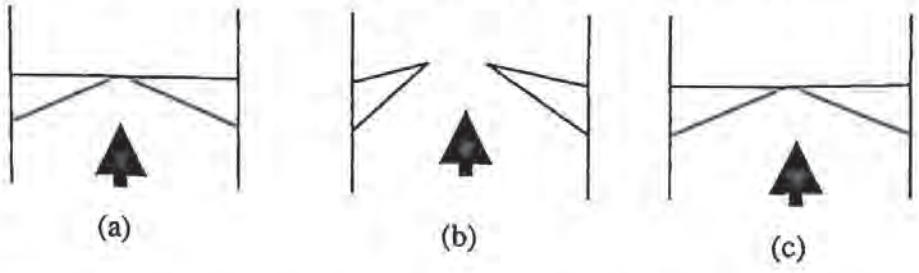


Figure 2.6 Vocal fold cycling at the larynx. (a) Closed with sub-glottal pressure buildup; (b) trans-glottal pressure differential causing folds to blow apart; (c) pressure equalization and tissue elasticity forcing temporary reclosure of vocal folds, ready to begin next cycle.

The waveform of air pressure variations created by this process can be described as a periodic flow, in cubic centimeters per second (after [15]). As shown in Figure 2.7, during the time bracketed as *one cycle*, there is no air flow during the initial closed portion. Then as

the glottis opens (open phase), the volume of air flow becomes greater. After a short peak, the folds begin to resume their original position and the air flow declines until complete closure is attained, beginning the next cycle. A common measure is the number of such cycles per second (Hz), or the fundamental frequency (*F0*). Thus the fundamental frequency for the waveform in Figure 2.7 is about 120 Hz.

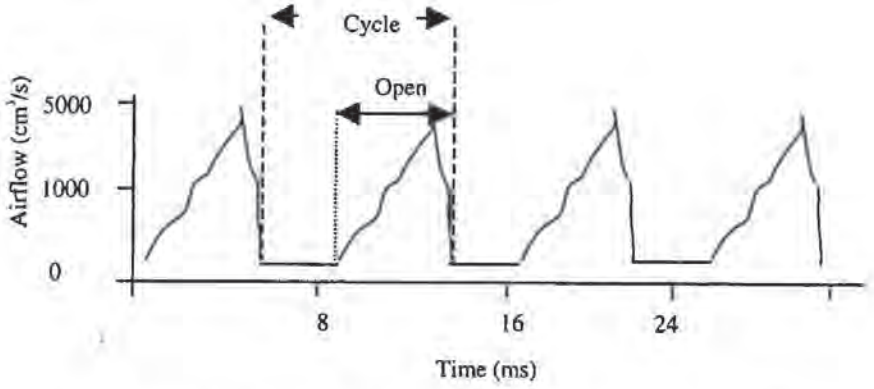


Figure 2.7 Waveform showing air flow during laryngeal cycle.

2.1.2.3. Spectrograms and Formants

Since the glottal wave is periodic, consisting of fundamental frequency (*F0*) and a number of harmonics (integral multiples of *F0*), it can be analyzed as a sum of sine waves as discussed in Chapter 5. The resonances of the vocal tract (above the glottis) are excited by the glottal energy. Suppose, for simplicity, we regard the vocal tract as a straight tube of uniform cross-sectional area, closed at the glottal end, open at the lips. When the shape of the vocal tract changes, the resonances change also. Harmonics near the resonances are emphasized, and, in speech, the resonances of the cavities that are typical of particular articulator configurations (e.g., the different vowel timbres) are called *formants*. The vowels in an actual speech waveform can be viewed from a number of different perspectives, emphasizing either a *cross-sectional* view of the harmonic responses at a single moment, or a longer-term view of the formant track evolution over time. The actual spectral analysis of a vowel at a single time-point, as shown in Figure 2.8, gives an idea of the uneven distribution of energy in resonances for the vowel */iy/* in the waveform for *see*, which is shown in Figure 2.5.

Another view of *sees* of Figure 2.5, called a spectrogram, is displayed in the lower part of Figure 2.9. It shows a long-term frequency analysis, comparable to a complete series of single time-point *cross sections* (such as that in Figure 2.8) ranged alongside one another in time and viewed from *above*.

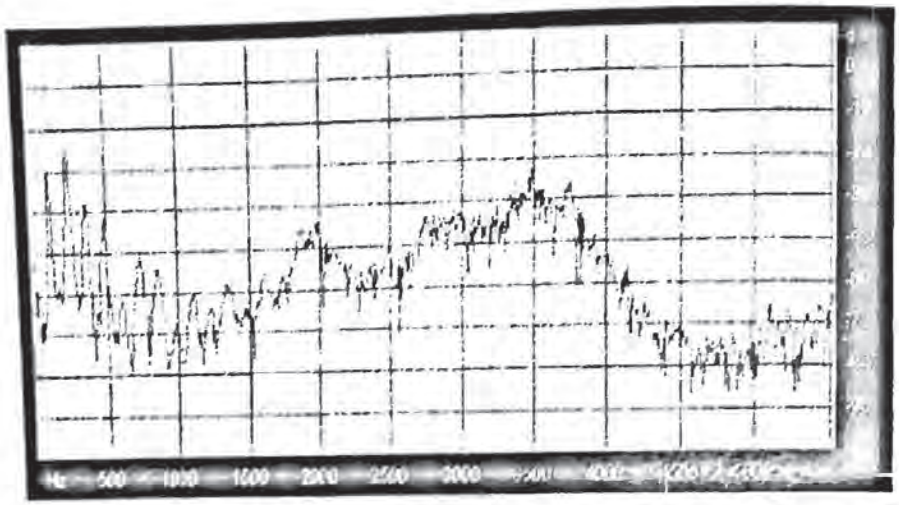


Figure 2.8 A spectral analysis of the vowel */iy/*, showing characteristically uneven distribution of energy at different frequencies.

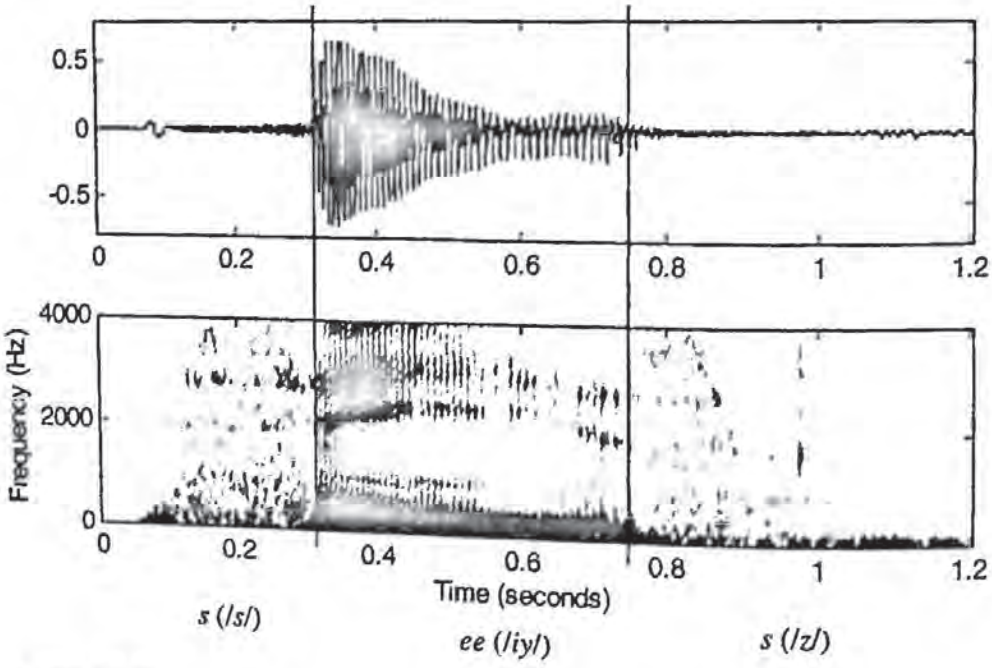


Figure 2.9 The spectrogram representation of the speech waveform *sees* (approximate phone boundaries are indicated with heavy vertical lines).

In the spectrogram of Figure 2.9, the darkness or lightness of a band indicates the relative amplitude or energy present at a given frequency. The dark horizontal bands show the formants, which are harmonics of the fundamental at natural resonances of the vocal tract cavity position for the vowel */iy/* in *see*. The mathematical methods for deriving analyses and representations such as those illustrated above are covered in Chapters 5 and 6.

2.1.3. Speech Perception

There are two major components in the auditory perception system: the peripheral auditory organs (ears) and the auditory nervous system (brain). The ear processes an acoustic pressure signal by first transforming it into a mechanical vibration pattern on the basilar membrane, and then representing the pattern by a series of pulses to be transmitted by the auditory nerve. Perceptual information is extracted at various stages of the auditory nervous system. In this section we focus mainly on the auditory organs.

2.1.3.1. Physiology of the Ear

The human ear, as shown in Figure 2.10, has three sections: the outer ear, the middle ear, and the inner ear. The outer ear consists of the external visible part and the external auditory canal that forms a tube along which sound travels. This tube is about 2.5 cm long and is covered by the eardrum at the far end. When air pressure variations reach the eardrum from the outside, it vibrates, and transmits the vibrations to bones adjacent to its opposite side. The vibration of the eardrum is at the same frequency (alternating compression and rarefaction) as the incoming sound pressure wave. The middle ear is an air-filled space or cavity about 1.3 cm across, and about 6 cm^3 volume. The air travels to the middle ear cavity along the tube (when opened) that connects the cavity with the nose and throat. The oval window shown in Figure 2.10 is a small membrane at the bony interface to the inner ear (cochlea). Since the cochlear walls are bony, the energy is transferred by mechanical action of the stapes into an impression on the membrane stretching over the oval window.

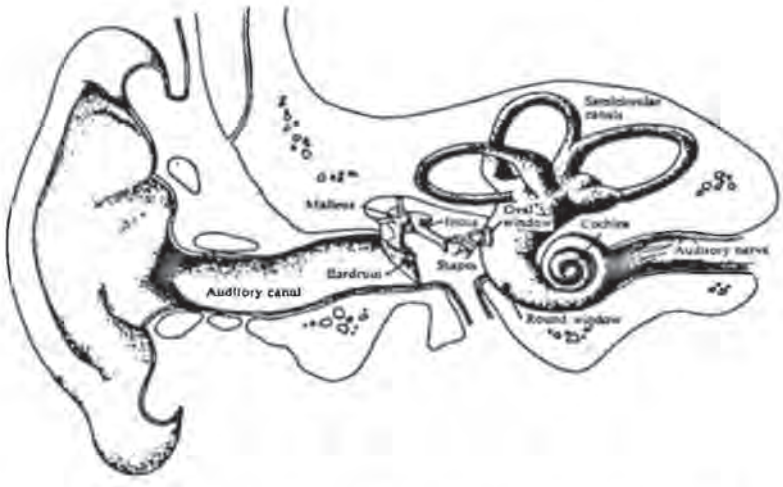


Figure 2.10 The structure of the peripheral auditory system with the outer, middle, and inner ear.

The relevant structure of the inner ear for sound perception is the cochlea, which communicates directly with the auditory nerve, conducting a representation of sound to the brain. The cochlea is a spiral tube about 3.5 cm long, which coils about 2.6 times. The spiral is divided, primarily by the basilar membrane running lengthwise, into two fluid-filled chambers. The cochlea can be roughly regarded as a filter bank, whose outputs are ordered by location, so that a frequency-to-place transformation is accomplished. The filters closest to the cochlear base respond to the higher frequencies, and those closest to its apex respond to the lower.

2.1.3.2. Physical vs. Perceptual Attributes

In psychoacoustics, a basic distinction is made between the perceptual attributes of a sound, especially a speech sound, and the measurable physical properties that characterize it. Each of the perceptual attributes, as listed in Table 2.2, seems to have a strong correlation with one main physical property, but the connection is complex, because other physical properties of the sound may affect perception in complex ways.

Table 2.2 Relation between perceptual and physical attributes of sound.

Physical Quantity	Perceptual Quality
Intensity	Loudness
Fundamental frequency	Pitch
Spectral shape	Timbre
Onset/offset time	Timing
Phase difference in binaural hearing	Location

Although sounds with a greater intensity level usually sound louder, the sensitivity of the ear varies with the frequency and the quality of the sound. One fundamental divergence between physical and perceptual qualities is the phenomenon of non-uniform *equal loudness* perception of tones of varying frequencies. In general, tones of differing pitch have different inherent *perceived loudness*. The sensitivity of the ear varies with the frequency and the quality of the sound. The graph of equal loudness contours adopted by ISO is shown in Figure 2.11. These curves demonstrate the relative insensitivity of the ear to sounds of low frequency at moderate to low intensity levels. Hearing sensitivity reaches a maximum around 4000 Hz, which is near the first resonance frequency of the outer ear canal, and peaks again around 13 kHz, the frequency of the second resonance [38].

Pitch is indeed most closely related to the fundamental frequency. The higher the fundamental frequency, the higher the pitch we perceive. However, discrimination between two pitches depends on the frequency of the lower pitch. Perceived pitch will change as intensity is increased and frequency is kept constant.

In another example of the non-identity of acoustic and perceptual effects, it has been observed experimentally that when the ear is exposed to two or more different tones, it is a common experience that one tone may *mask* the others. Masking is probably best explained

as an upward shift in the hearing threshold of the weaker tone by the louder tone. Pure tones, complex sounds, narrow and broad bands of noise all show differences in their ability to mask other sounds. In general, pure tones close together in frequency mask each other more than tones widely separated in frequency. A pure tone masks tones of higher frequency more effectively than tones of lower frequency. The greater the intensity of the masking tone, the broader the range of the frequencies it can mask [18, 31].

Binaural listening greatly enhances our ability to sense the direction of the sound source. The sense of localization attention is mostly focused on side-to-side discrimination or *lateralization*. Time and intensity cues have different impacts for low frequency and high frequency, respectively. Low-frequency sounds are lateralized mainly on the basis of interaural time difference, whereas high-frequency sounds are localized mainly on the basis of interaural intensity differences [5].

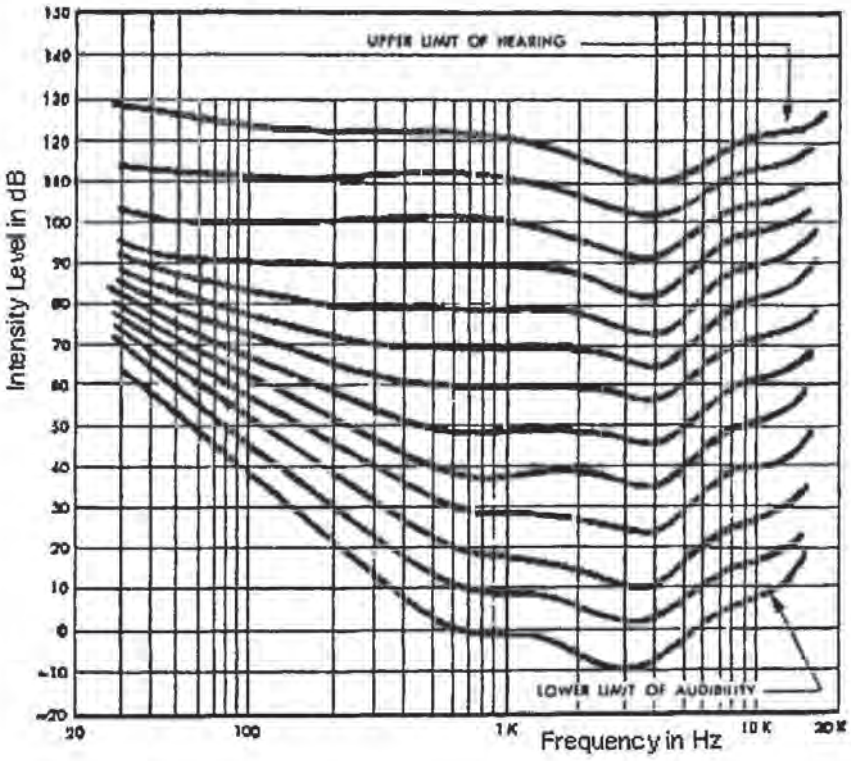


Figure 2.11 Equal-loudness curves indicate that the response of the human hearing mechanism is a function of frequency and loudness levels. This relationship again illustrates the difference between physical dimensions and psychological experience (after ISO 226).

Finally, an interesting perceptual issue is the question of distinctive voice quality. Speech from different people sounds different. Partially this is due to obvious factors, such as differences in characteristic fundamental frequency caused by, for example, the greater mass and length of adult male vocal folds as opposed to female. But there are more subtle

effects as well. In psychoacoustics, the concept of *timbre* (of a sound or instrument) is defined as that attribute of auditory sensation by which a subject can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar. In other words, when all the easily measured differences are controlled, the remaining perception of difference is ascribed to timbre. This is heard most easily in music, where the same note in the same octave played for the same duration on a violin sounds different from a flute. The timbre of a sound depends on many physical variables including a sound's spectral power distribution, its temporal envelope, rate and depth of amplitude or frequency modulation, and the degree of inharmonicity of its harmonics.

2.1.3.3. Frequency Analysis

Researchers have undertaken psychoacoustic experimental work to derive frequency scales that attempt to model the natural response of the human perceptual system, since the cochlea of the inner ear acts as a spectrum analyzer. The complex mechanism of the inner ear and auditory nerve implies that the perceptual attributes of sounds at different frequencies may not be entirely simple or linear in nature. It is well known that the western musical pitch is described in *octaves*[1] and *semi-tones*.[2] The perceived musical pitch of complex tones is basically proportional to the logarithm of frequency. For complex tones, the just noticeable difference for frequency is essentially constant on the octave/semi-tone scale. Musical pitch scales are used in prosodic research (on speech intonation contour generation).

AT&T Bell Labs has contributed many influential discoveries in hearing, such as critical band and articulation index, since the turn of the 20th century [3]. Fletcher's work [14] pointed to the existence of critical bands in the cochlear response. Critical bands are of great importance in understanding many auditory phenomena such as perception of loudness, pitch, and timbre. The auditory system performs frequency analysis of sounds into their component frequencies. The cochlea acts as if it were made up of overlapping filters having bandwidths equal to the critical bandwidth. One class of critical band scales is called *Bark frequency scale*. It is hoped that by treating spectral energy over the Bark scale, a more natural fit with spectral information processing in the ear can be achieved. The Bark scale ranges from 1 to 24 Barks, corresponding to 24 critical bands of hearing as shown in Table 2.3. As shown in Figure 2.12, the perceptual resolution is finer in the lower frequencies. It should be noted that the ear's critical bands are continuous, and a tone of any audible frequency always finds a critical band centered on it. The Bark frequency b can be expressed in terms of the linear frequency (in Hz) by

$$b(f) = 13\arctan(0.00076f) + 3.5 * \arctan\left((f/7500)^2\right) \quad (Bark) \tag{2.5}$$

[1] A tone of frequency f_1 is said to be an octave above a tone with frequency f_2 if and only if $f_1 = 2f_2$.
[2] There are 12 semitones in one octave, so a tone of frequency f_1 is said to be a semitone above a tone with frequency f_2 if and only if $f_1 = 2^{1/12} f_2 = 1.05946 f_2$.

Table 2.3 The Bark frequency scale.

Bark Band #	Edge (Hz)	Center (Hz)
1	100	50
2	200	150
3	300	250
4	400	350
5	510	450
6	630	570
7	770	700
8	920	840
9	1080	1000
10	1270	1170
11	1480	1370
12	1720	1600
13	2000	1850
14	2320	2150
15	2700	2500
16	3150	2900
17	3700	3400
18	4400	4000
19	5300	4800
20	6400	5800
21	7700	7000
22	9500	8500
23	12000	10500
24	15500	13500

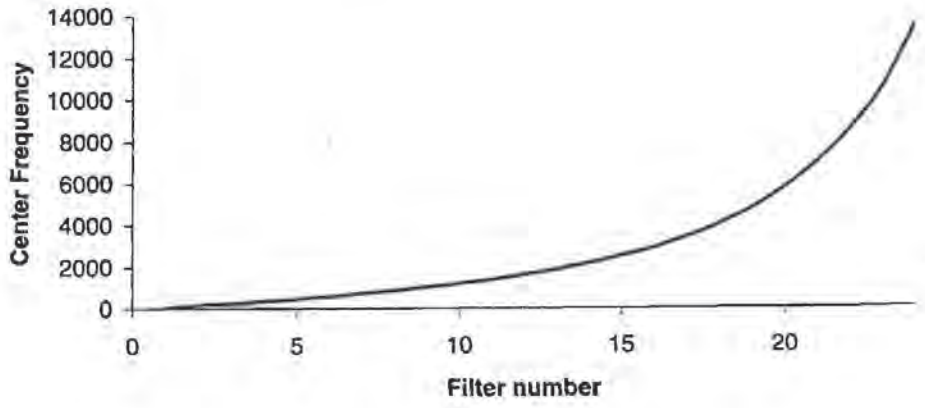


Figure 2.12 The center frequency of 24 Bark frequency filters as illustrated in Table 2.3.

Another such perceptually motivated scale is the mel frequency scale [41], which is linear below 1 kHz, and logarithmic above, with equal numbers of samples taken below and above 1 kHz. The mel scale is based on experiments with simple tones (sinusoids) in which subjects were required to divide given frequency ranges into four perceptually equal intervals or to adjust the frequency of a stimulus tone to be half as high as that of a comparison tone. One mel is defined as one thousandth of the pitch of a 1 kHz tone. As with all such attempts, it is hoped that the mel scale more closely models the sensitivity of the human ear than a purely linear scale and provides for greater discriminatory capability between speech segments. Mel-scale frequency analysis has been widely used in modern speech recognition systems. It can be approximated by:

$$B(f) = 1125 \ln(1 + f/700) \tag{2.6}$$

The mel scale is plotted in Figure 2.13 together with the Bark scale and the bilinear transform (see Chapter 6).

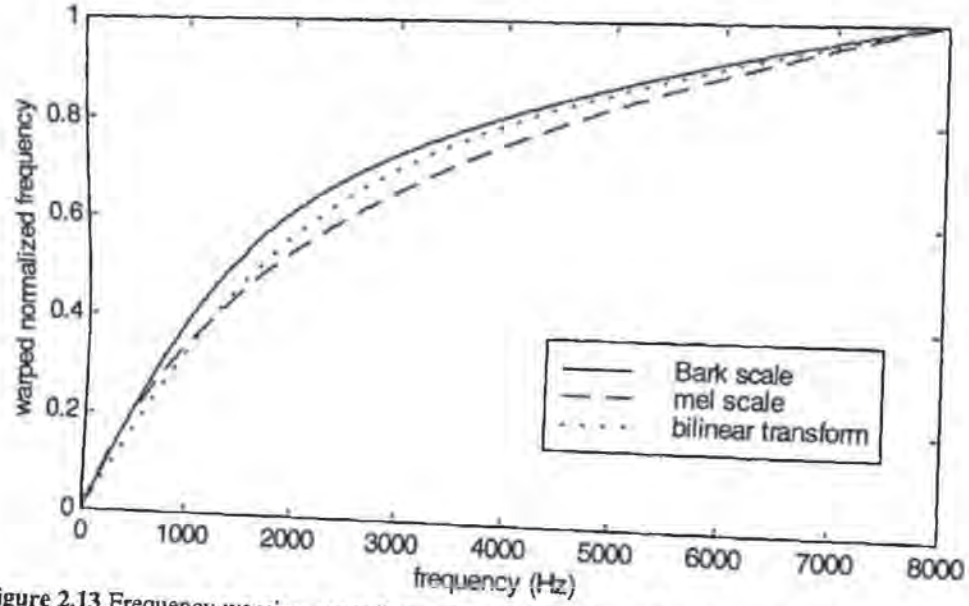


Figure 2.13 Frequency warping according to the Bark scale, ERB scale, mel-scale, and bilinear transform for $\alpha = 0.6$: linear frequency in the x-axis and normalized frequency in the y-axis.

A number of techniques in the modern spoken language system, such as cepstral analysis, and dynamic feature, have benefited tremendously from perceptual research as discussed throughout this book.

2.1.3.4. Masking

Frequency masking is a phenomenon under which one sound cannot be perceived if another sound close in frequency has a high enough level. The first sound *masks* the other one. Fre-

quency-masking levels have been determined empirically, with complicated models that take into account whether the masker is a tone or noise, the masker's level, and other considerations.

We now describe a phenomenon known as *tone-masking noise*. It has been determined empirically that noise with energy E_N (dB) at Bark frequency g masks a tone at Bark frequency b if the tone's energy is below the threshold

$$T_T(b) = E_N - 6.025 - 0.275g + S_m(b-g) \quad (dB\ SPL) \tag{2.7}$$

where the *spread-of-masking* function $S_m(b)$ is given by

$$S_m(b) = 15.81 + 7.5(b+0.474) - 17.5\sqrt{1+(b+0.474)^2} \quad (dB) \tag{2.8}$$

We now describe a phenomenon known as *noise-masking tone*. It has been determined empirically that a tone at Bark frequency g with energy E_T (dB) masks noise at Bark frequency b if the noise energy is below the threshold

$$T_N(b) = E_T - 2.025 - 0.175g + S_m(b-g) \quad (dB\ SPL) \tag{2.9}$$

Masking thresholds are commonly referred to in the literature as Bark scale functions of *just noticeable distortion* (JND). Equation (2.8) can be approximated by a triangular spreading function that has slopes of +25 and –10 dB per Bark, as shown in Figure 2.14.

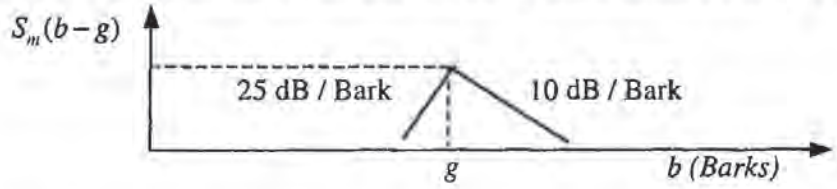


Figure 2.14 Contribution of Bark frequency g to the masked threshold $S_m(b)$.

In Figure 2.15 we show both the threshold of hearing and the masked threshold of a tone at 1 kHz with a 69 dB SPL. The combined masked threshold is the sum of the two in the linear domain

$$T(f) = 10\log_{10}\left(10^{0.1T_h(f)} + 10^{0.1T_T(f)}\right) \tag{2.10}$$

which is approximately the largest of the two.

In addition to frequency masking, there is a phenomenon called temporal masking by which a sound too close in time to another sound cannot be perceived. Whereas premasking tends to last about 5 ms, postmasking can last from 50 to 300 ms. Temporal masking level of a masker with a uniform level starting at 0 ms and lasting 200 ms is shown in Figure 2.16.

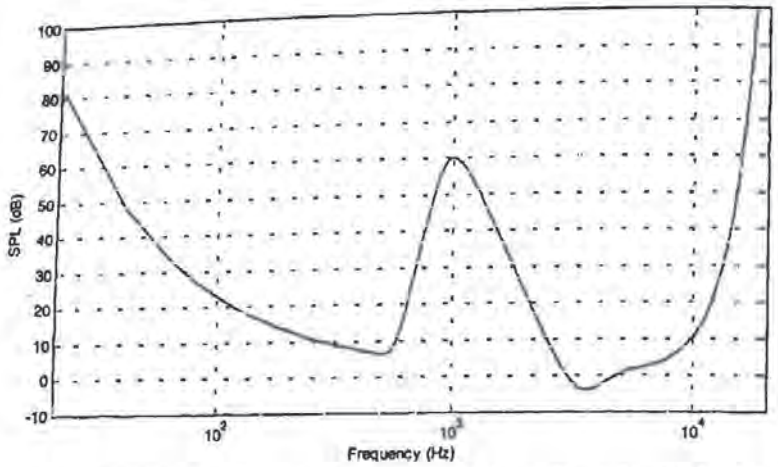


Figure 2.15 Absolute threshold of hearing and spread of masking threshold for a 1 kHz sine-wave masker with a 69 dB SPL. The overall masked threshold is approximately the largest of the two thresholds.

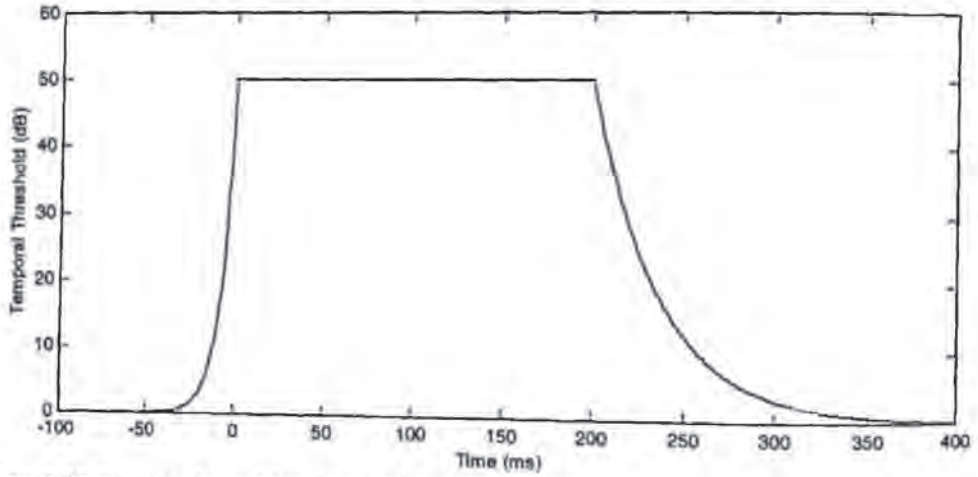


Figure 2.16 Temporal masking level of a masker with a uniform level starting at 0 ms and lasting 200 ms.

2.2. PHONETICS AND PHONOLOGY

We now discuss basic phonetics and phonology needed for spoken language processing. *Phonetics* refers to the study of speech sounds and their production, classification, and transcription. *Phonology* is the study of the distribution and patterning of speech sounds in a language and of the tacit rules governing pronunciation.

2.2.1. Phonemes

Linguist Ferdinand de Saussere (1857-1913) is credited with the observation that the relation between a sign and the object signified by it is arbitrary. The same concept, a certain yellow and black flying social insect, has the sign *honeybee* in English and *mitsubachi* in Japanese.

There is no particular relation between the various pronunciations and the meaning, nor do these pronunciations per se describe the bee's characteristics in any detail. For phonetics, this means that the speech sounds described in this chapter have no inherent meaning, and should be randomly distributed across the lexicon, except as affected by extraneous historical or etymological considerations. The sounds are just a set of arbitrary effects made available by human vocal anatomy. You might wonder about this theory when you observe, for example, the number of words beginning with *sn* that have to do with nasal functions in English: *sneeze, snort, sniff, snot, snore, snuffle*, etc. But Saussere's observation is generally true, except for obvious onomatopoetic (sound) words like *buzz*.

Like fingerprints, every speaker's vocal anatomy is unique, and this makes for unique vocalizations of speech sounds. Yet language communication is based on commonality of form at the perceptual level. To allow discussion of the commonalities, researchers have identified certain gross characteristics of speech sounds that are adequate for description and classification of words in dictionaries. They have also adopted various systems of notation to represent the subset of phonetic phenomena that are crucial for meaning.

As an analogy, consider the system of computer coding of text characters. In such systems, the *character* is an abstraction, e.g. the Unicode character U+0041. The identifying property of this character is its Unicode name *LATIN CAPITAL LETTER A*. This is a genuine abstraction; no particular realization is necessarily specified. As the Unicode 2.1 standard [1] states:

The Unicode Standard does not define glyph images. The standard defines how characters are interpreted, not how glyphs are rendered. The software or hardware-rendering engine of a computer is responsible for the appearance of the characters on the screen. The Unicode Standard does not specify the size, shape, nor orientation of on-screen characters.

Thus, the U+0041 character can be realized differently for different purposes, and in different sizes with different fonts:

U+0041 → A, *A*, A, A, A, ...

The realizations of the character U+0041 are called glyphs, and there is no distinguished uniquely correct glyph for U+0041. In speech science, the term *phoneme* is used to denote any of the minimal units of speech sound in a language that can serve to distinguish one word from another. We conventionally use the term *phone* to denote a phoneme's acoustic realization. In the example given above, U+0041 corresponds to a phoneme and the various fonts correspond to the phone. For example, English phoneme */t/* have two very different acoustic realizations in the words *sat* and *meter*. You had better treat them as two different phones if you want to build a spoken language system. We will use the terms *phone* or *phoneme* interchangeably to refer to the speaker-independent and context-independent units of meaningful sound contrast. Table 2.4 shows a complete list of phonemes used in American English. The set of phonemes will differ in realization across individual speakers. But phonemes will always function systematically to differentiate meaning in words, just as the phoneme */p/* signals the word *pat* as opposed to the similar-sounding but distinct *bat*. The important contrast distinguishing this pair of words is */p/* vs. */b/*.

In this section we concentrate on the basic qualities that *define and differentiate abstract phonemes*. In Section 2.2.1.3 below we consider *why and how phonemes vary* in their actual realizations by different speakers and in different contexts.

Table 2.4 English phonemes used for typical spoken language systems.

Phonemes	Word Examples	Description
iy	*feel, eve, me*	front close unrounded
ih	*fill, hit, lid*	front close unrounded (lax)
ae	*at, carry, gas*	front open unrounded (tense)
aa	*father, ah, car*	back open unrounded
ah	*cut, bud, up*	open-mid back unrounded
ao	*dog, lawn, caught*	open-mid back round
ay	*tie, ice, bite*	diphthong with quality: aa + ih
ax	*ago, comply*	central close mid (schwa)
ey	*ate, day, tape*	front close-mid unrounded (tense)
eh	*pet, berry, ten*	front open-mid unrounded
er	*turn, fur, meter*	central open-mid unrounded rhoti-
ow	*go, own, tone*	back close-mid rounded
aw	*foul, how, our*	diphthong with quality: aa + uh
oy	*toy, coin, oil*	diphthong with quality: ao + ih
uh	*book, pull, good*	back close-mid unrounded (lax)
uw	*tool, crew, moo*	back close round
b	*big, able, tab*	voiced bilabial plosive
p	*put, open, tap*	voiceless bilabial plosive
d	*dig, idea, wad*	voiced alveolar plosive
t	*talk, sat*	voiceless alveolar plosive &
t	*meter*	alveolar flap
g	*gut, angle, tag*	voiced velar plosive
k	*cut, ken, take*	voiceless velar plosive
f	*fork, after, if*	voiceless labiodental fricative
v	*vat, over, have*	voiced labiodental fricative
s	*sit, cast, toss*	voiceless alveolar fricative
z	*zap, lazy, haze*	voiced alveolar fricative
th	*thin, nothing, truth*	voiceless dental fricative
dh	*then, father, scythe*	voiced dental fricative
sh	*she, cushion, wash*	voiceless postalveolar fricative
zh	*genre, azure*	voiced postalveolar fricative
l	*lid*	alveolar lateral approximant
l	*elbow, sail*	velar lateral approximant
r	*red, part, far*	retroflex approximant
y	*yacht, yard*	palatal sonorant glide
w	*with, away*	labiovelar sonorant glide
hh	*help, ahead, hotel*	voiceless glottal fricative
m	*mat, amid, aim*	bilabial nasal
n	*no, end, pan*	alveolar nasal
ng	*sing, anger*	velar nasal
ch	*chin, archer, march*	voiceless alveolar affricate: t + sh
jh	*joy, agile, edge*	voiced alveolar affricate: d + zh

2.2.1.1. Vowels

The tongue shape and positioning in the oral cavity do not form a major constriction of air flow during vowel articulation. However, variations of tongue placement give each vowel its distinct character by changing the resonance, just as different sizes and shapes of bottles give rise to different acoustic effects when struck. The primary energy entering the pharyngeal and oral cavities in vowel production vibrates at the fundamental frequency. The major resonances of the oral and pharyngeal cavities for vowels are called F1 and F2 – the first and second formants, respectively. They are determined by tongue placement and oral tract shape in vowels, and they determine the characteristic timbre or quality of the vowel.

The relationship of F1 and F2 to one another can be used to describe the English vowels. While the shape of the complete vocal tract determines the spectral outcome in a complex, nonlinear fashion, generally F1 corresponds to the back or pharyngeal portion of the cavity, while F2 is determined more by the size and shape of the oral portion, forward of the major tongue extrusion. This makes intuitive sense – the cavity from the glottis to the tongue extrusion is longer than the forward part of the oral cavity, thus we would expect its resonance to be lower. In the vowel of *see*, for example, the tongue extrusion is far forward in the mouth, creating an exceptionally long rear cavity, and correspondingly low F1. The forward part of the oral cavity, at the same time, is extremely short, contributing to higher F2. This accounts for the wide separation of the two lowest dark horizontal bands in Figure 2.9, corresponding to F1 and F2, respectively. Rounding the lips has the effect of extending the front-of-tongue cavity, thus lowering F2. Typical values of F1 and F2 of American English vowels are listed in Table 2.5.

Table 2.5 Phoneme labels and typical formant values for vowels of English.

Vowel Labels	Mean F1 (Hz)	Mean F2 (Hz)
iy (feel)	300	2300
ih (fill)	360	2100
ae (gas)	750	1750
aa (father)	680	1100
ah (cut)	720	1240
ao (dog)	600	900
ax (comply)	720	1240
eh (pet)	570	1970
er (turn)	580	1380
ow (tone)	600	900
uh (good)	380	950
uw (tool)	300	940

The characteristic F1 and F2 values for vowels are sometimes called formant targets, which are ideal locations for perception. Sometimes, due to fast speaking or other limitations on performance, the speaker cannot quite attain an ideal target before the articulators begin shifting to targets for the following phoneme, which is phonetic context dependent. Additionally, there is a special class of vowels that combine two distinct sets of F1/F2 targets.

These are called *diphthongs*. As the articulators move, the initial vowel targets glide smoothly to the final configuration. Since the articulators are working faster in production of a diphthong, sometimes the *ideal* formant target values of the component values are not fully attained. Typical diphthongs of American English are listed in Table 2.6.

Table 2.6 The diphthongs of English.

Diphthong Labels	Components
ay (tie)	/aa/ → /iy/
ey (ate)	/eh/ → /iy/
oy (coin)	/ao/ → /iy/
aw (foul)	/aa/ → /uw/

Figure 2.17 shows the first two formants for a number of typical vowels.

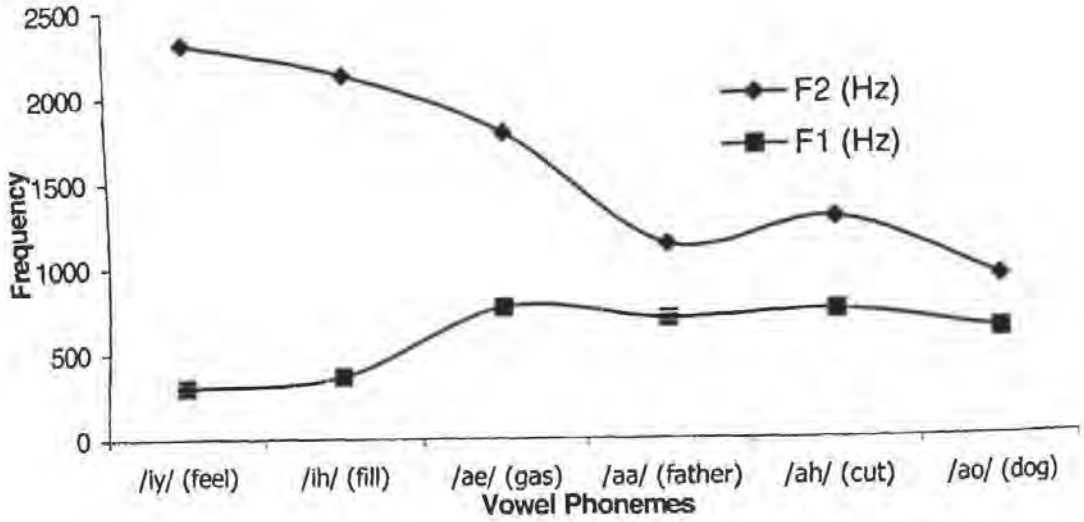


Figure 2.17 F1 and F2 values for articulations of some English vowels.

The major articulator for English vowels is the middle to rear portion of the tongue. The position of the tongue's surface is manipulated by large and powerful muscles in its root, which move it as a whole within the mouth. The linguistically important dimensions of movement are generally the ranges [front ⇔ back] and [high ⇔ low]. You can feel this movement easily. Say mentally, or whisper, the sound */iy/* (as in *see*) and then */aa/* (as in *father*). Do it repeatedly, and you will get a clear perception of the tongue movement from high to low. Now try */iy/* and then */uw/* (as in *blue*), repeating a few times. You will get a clear perception of place of articulation from front */iy/* to back */uw/*. Figure 2.18 shows a schematic characterization of English vowels in terms of relative tongue positions. There are two kinds of vowels: those in which tongue height is represented as a point and those in which it is represented as a vector.

Though the tongue hump is the major actor in vowel articulation, other articulators come into play as well. The most important secondary vowel mechanism for English and many other languages is lip rounding. Repeat the exercise above, moving from the */iy/* (*see*)

to the */uw/* (*blue*) position. Now rather than noticing the tongue movement, pay attention to your lip shape. When you say */iy/*, your lips will be flat, slightly open, and somewhat spread. As you move to */uw/*, they begin to *round out*, ending in a more puckered position. This lengthens the oral cavity during */uw/*, and affects the spectrum in other ways.

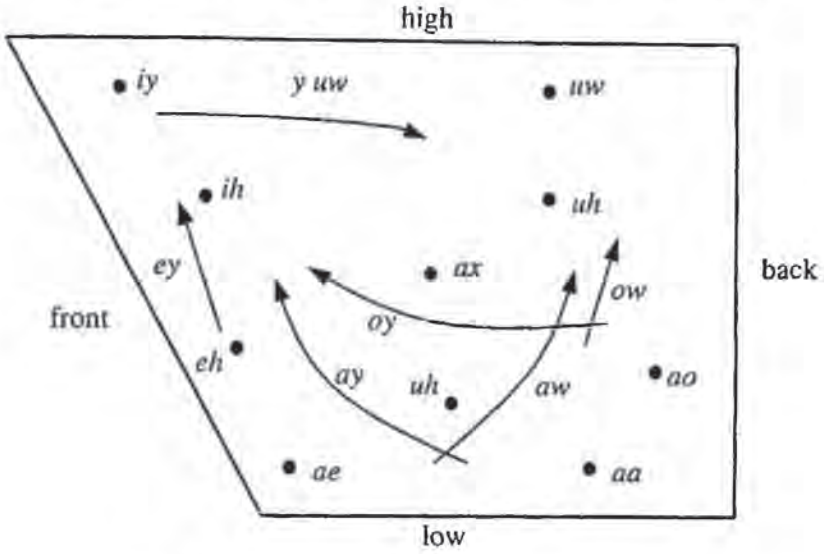


Figure 2.18 Relative tongue positions of English vowels [24].

Though there is always some controversy, linguistic study of phonetic abstractions, called *phonology*, has largely converged on the five binary features: +/- high, +/- low, +/- front, +/- back, and +/- round, plus the phonetically ambiguous but phonologically useful feature +/- tense, as adequate to uniquely characterize the major vowel distinctions of Standard English (and many other languages). Obviously, such a system is a little bit too free with logically contradictory specifications, such as [+high, +low], but these are excluded from real-world use. These features can be seen in Table 2.7.

Table 2.7 Phonological (abstract) feature decomposition of basic English vowels.

Vowel	high	low	front	back	round	tense
iy	+	-	+	-	-	+
ih	+	-	+	-	-	-
ae	-	+	+	-	-	+
aa	-	+	-	-	-	+
ah	-	-	-	-	-	+
ao	-	+	-	+	+	+
ax	-	-	-	-	-	-
eh	-	-	+	-	-	-
ow	-	-	-	+	+	+
uh	+	-	-	+	-	-
uw	+	-	-	+	-	+

This kind of abstract analysis allows researchers to make convenient statements about classes of vowels that behave similarly under certain conditions. For example, one may speak simply of the high vowels to indicate the set */iy, ih, uh, uw/*.

2.2.1.2. Consonants

Consonants, as opposed to vowels, are characterized by significant constriction or obstruction in the pharyngeal and/or oral cavities. Some consonants are voiced; others are not. Many consonants occur in pairs, that is, they share the same configuration of articulators, and one member of the pair additionally has voicing which the other lacks. One such pair is */s, z/*, and the voicing property that distinguishes them shows up in the non-periodic noise of the initial segment */s/* in Figure 2.5 as opposed to the voiced consonant end-phone, */z/*. Manner of articulation refers to the articulation mechanism of a consonant. The major distinctions in manner of articulation are listed in Table 2.8.

Table 2.8 Consonant manner of articulation.

Manner	Sample Phone	Example Words	Mechanism
Plosive	*/p/*	tat, tap	Closure in oral cavity
Nasal	*/m/*	team, meet	Closure of nasal cavity
Fricative	*/s/*	sick, kiss	Turbulent airstream noise
Retroflex liquid	*/r/*	rat, tar	Vowel-like, tongue high and curled back
Lateral liquid	*/l/*	lean, kneel	Vowel-like, tongue central, side airstream
Glide	*/y/,/w/*	yes, well	Vowel-like

The English phones that typically have voicing without complete obstruction or narrowing of the vocal tract are called *semivowels* and include */l, r/*, the *liquid* group, and */y, w/*, the *glide* group. Liquids, glides, and vowels are all *sonorant*, meaning they have continuous voicing. Liquids */l/* and */r/* are quite vowel-like and in fact may become *syllabic* or act entirely as vowels in certain positions, such as the *l* at the end of *edible*. In */l/*, the airstream flows around the sides of the tongue, leading to the descriptive term *lateral*. In */r/*, the tip of the tongue is curled back slightly, leading to the descriptive term *retroflex*. Figure 2.19 shows some semivowels.

Glides */y, w/* are basically vowels */iy, uw/* whose initial position within the syllable require them to be a little shorter and to lack the ability to be stressed, rendering them just different enough from true vowels that they are classed as a special category of consonant. Pre-vocalic glides that share the syllable-initial position with another consonant, such as the */y/* in the second syllable of *computer /k uh m . p y uw . t er/*, or the */w/* in *quick /k w ih k/*, are sometimes called *on-glides*. The semivowels, as a class, are sometimes called *approximants*, meaning that the tongue approaches the top of the oral cavity, but does not completely contact so as to obstruct the air flow.

Even the non-sonorant consonants that require complete or close-to-complete obstruction may still maintain some voicing before or during the obstruction, until the pressure dif-

ferential across the glottis starts to disappear, due to the closure. Such voiced consonants include */b,d,g, z, zh, v/*. They have a set of counterparts that differ only in their characteristic lack of voicing: */p,t,k, s, sh, f/*.

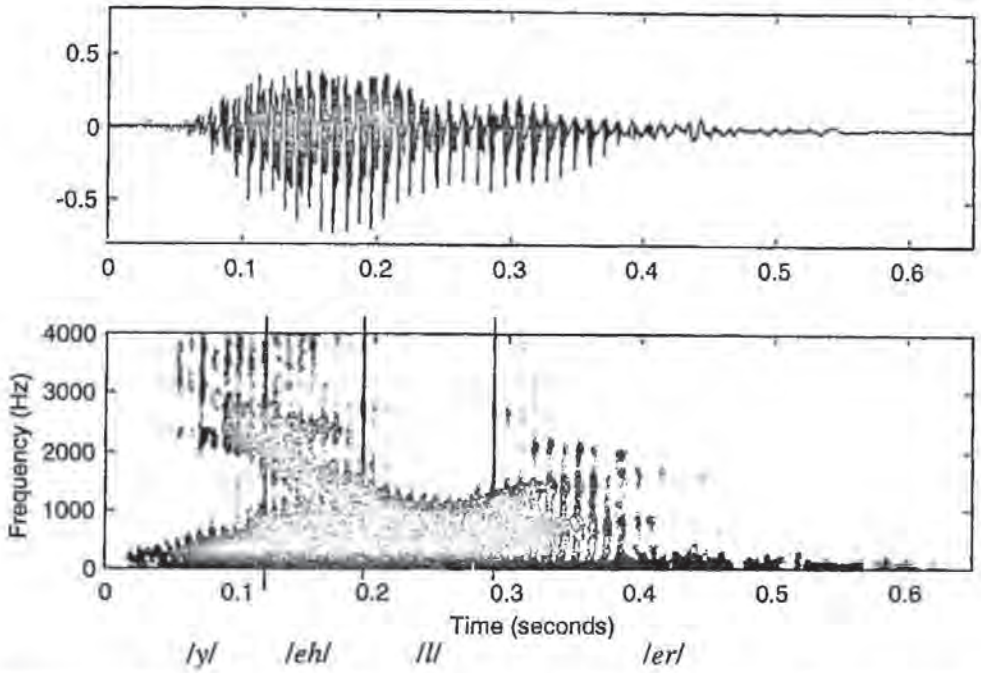


Figure 2.19 Spectrogram for the word *yeller*, showing semivowels */y/*, */l/*, */er/* (approximate phone boundaries shown with vertical lines).

Nasal consonants */m,n/* are a mixed bag: the oral cavity has significant constriction (by the tongue or lips), yet the voicing is continuous, like that of the sonorants, because, with the velar flap open, air passes freely through the nasal cavity, maintaining a pressure differential across the glottis.

A consonant that involves complete blockage of the oral cavity is called an obstruent stop, or plosive consonant. These may be voiced throughout if the trans-glottal pressure drop can be maintained long enough, perhaps through expansion of the wall of the oral cavity. In any case, there can be voicing for the early sections of stops. Voiced, unvoiced pairs of stops include: */b,p/*, */d,t/*, and */g,k/*. In viewing the waveform of a stop, a period of silence corresponding to the oral closure can generally be observed. When the closure is removed (by opening the constrictor, which may be lips or tongue), the trapped air rushes out in a more or less sudden manner. When the upper oral cavity is unimpeded, the closure of the vocal folds themselves can act as the initial blocking mechanism for a type of stop heard at the very beginning of vowel articulation in vowel-initial words like *atrophy*. This is called a *glottal stop*. Voiceless plosive consonants in particular exhibit a characteristic aperiodic *burst* of energy at the (articulatory) point of closure as shown in Figure 2.20 just prior to */i/*. By com-

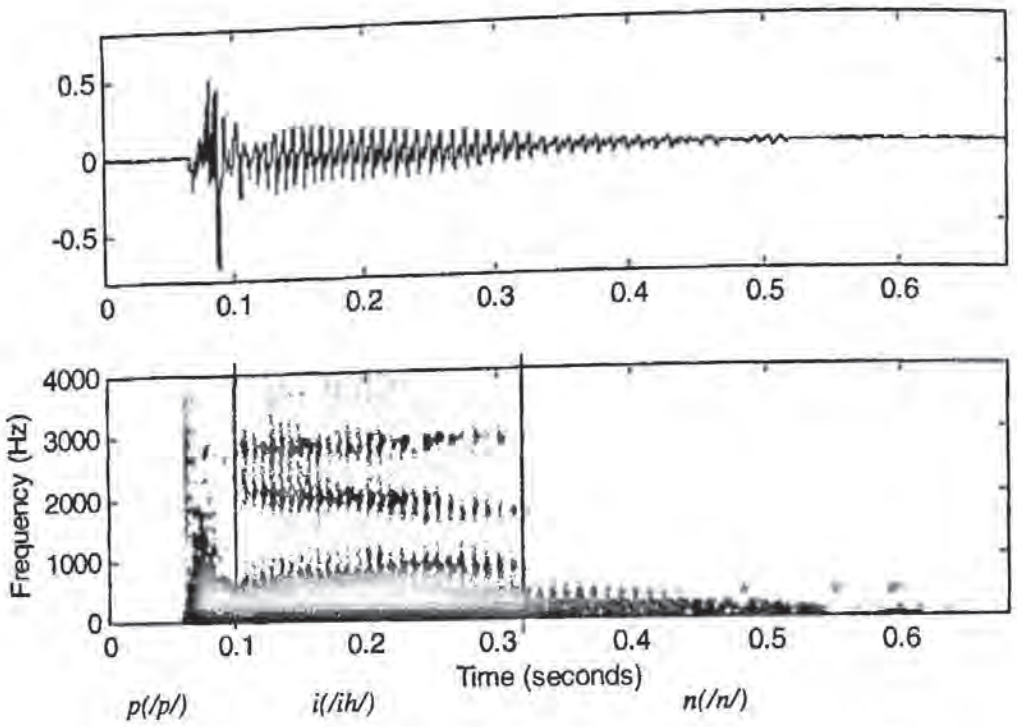


Figure 2.20 Spectrogram: stop release *burst* of */p/* in the word *pin*.

parison, the voicing of voiced plosive consonants may not always be obvious in a spectrogram.

A consonant that involves nearly complete blockage of some position in the oral cavity creates a narrow stream of turbulent air. The friction of this air stream creates a nonperiodic hiss-like effect. Sounds with this property are called fricatives and include */s, z/*. There is no voicing during the production of *s*, while there can be voicing (in addition to the frication noise), during the production of *z*, as discussed above. */s, z/* have a common place of articulation, as explained below, and thus form a natural similarity class. Though controversial, */h/* can also be thought of as a (glottal) fricative. */s/* in word-initial position and */z/* in word-final position are exemplified in Figure 2.5.

Some sounds are complex combinations of manners of articulation. For example, the *affricates* consist of a stop (e.g., */t/*), followed by a fricative [e.g., */sh/*) combining to make a unified sound with rapid phases of closure and continuancy (e.g., $\{t + sh\} = ch$ as in *church*). The affricates in English are the voiced/unvoiced pairs: */j/* $(d + zh)$ and */ch/* $(t + sh)$. The complete consonant inventory of English is shown in Table 2.9.

Consider the set */m/*, */n/*, */ng/* from Table 2.9. They are all voiced nasal consonants, yet they sound distinct to us. The difference lies in the location of the major constriction along the top of the oral cavity (from lips to velar area) that gives each consonant its unique quality. The articulator used to touch or approximate the given location is usually some spot along the length of the tongue. As shown in Figure 2.21, the combination of articulator and place of articulation gives each consonant its characteristic sound:

Table 2.9 Manner of articulation of English consonants.

Consonant Labels	Consonant Examples	Voiced?	Manner
b	**b**ig, able, ta**b**	+	plosive
p	**p**ut, open, ta**p**	-	plosive
d	**d**ig, **d**ea, wa**d**	+	plosive
t	**t**alk, sa**t**	-	plosive
g	**g**ut, angle, ta**g**	+	plosive
k	**c**ut, oa**k**en, ta**k**e	-	plosive
v	**v**at, over, ha**v**e	+	fricative
f	**f**ork, after, i**f**	-	fricative
z	zap, lazy, ha**z**e	+	fricative
s	**s**it, ca**s**t, to**ss**	-	fricative
dh	**th**en, fa**th**er, scy**th**e	+	fricative
th	**th**in, no**th**ing, tru**th**	-	fricative
zh	genre, azure, beige	+	fricative
sh	**sh**e, cu**sh**ion, wa**sh**	-	fricative
jh	**j**oy, agile, e**dg**e	+	affricate
ch	**ch**in, ar**ch**er, ma**rch**	-	affricate
l	**l**id, el**b**ow, sa**il**	+	lateral
r	**r**ed, pa**r**t, fa**r**	+	retroflex
y	**y**acht, o**n**ion, **y**ard	+	glide
w	**w**ith, a**w**ay	+	glide
hh	**h**elp, a**h**ead, **h**otel	+	fricative
m	**m**at, a**m**id, ai**m**	+	nasal
n	**n**o, e**n**d, pa**n**	+	nasal
ng	sing, a**n**ger, dri**nk**	+	nasal

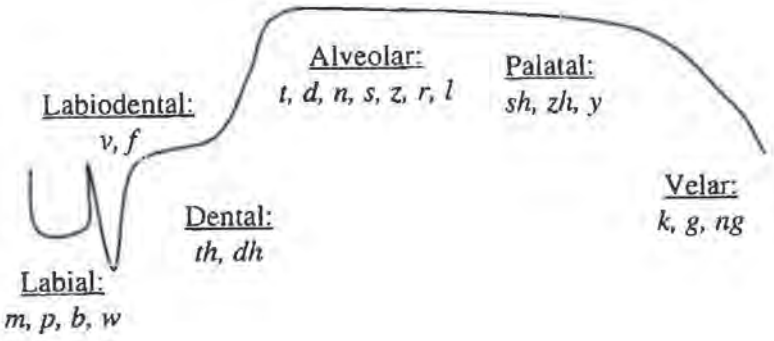


Figure 2.21 The major places of consonant articulation with respect to the human mouth.

- The *labial* consonants have their major constriction at the lips. This includes */p/, /b/* (these two differ only by manner of articulation) and */m/* and */w/*.
- The class of *dental or labio-dental* consonants includes */f, v/* and */th, dh/* (the members of these groups differ in manner, not place).
- *Alveolar* consonants bring the front part of the tongue, called the tip or the part behind the tip called the blade, into contact or approximation to the alveolar ridge, rising semi-vertically above and behind the teeth. These include */t, d, n, s, z, r, l/*. The members of this set again differ in manner of articulation (voicing, continuity, nasality), rather than place.
- *Palatal* consonants have approximation or constriction on or near the roof of the mouth, called the palate. The members include */sh, zh, y/*.
- *Velar* consonants bring the articulator (generally the back of the tongue), up to the rearmost top area of the oral cavity, near the velar flap. Velar consonants in English include */k, g/* (differing by voicing) and the nasal continuant */ng/*.

With the place terminology, we can complete the descriptive inventory of English consonants, arranged by manner (rows), place (columns), and voiceless/voiced (pairs in cells) as illustrated in Table 2.10.

Table 2.10 The consonants of English arranged by place (columns) and manner (rows).

	Labial	Labio-dental	Dental	Alveolar	Palatal	Velar	Glottal
Plosive	*p b*			*t d*		*k g*	*?*
Nasal	*m*			*n*		*ng*	
Fricative		*f v*	*th dh*	*s z*	*sh zh*		*h*
Retroflex sonorant				*r*			
Lateral sonorant				*l*			
Glide	*w*				*y*		

2.2.1.3. Phonetic Typology

The oral, nasal, pharyngeal, and glottal mechanisms actually make available a much wider range of effects than English happens to use. So, it is expected that other languages would utilize other vocal mechanisms, in an internally consistent but essentially arbitrary fashion, to represent their lexicons. In addition, often a vocal effect that is part of the systematic linguistic phonetics of one language is present in others in a less codified, but still perceptible, form. For example, Japanese vowels have a characteristic distinction of length that can be hard for non-natives to perceive and to use when learning the language. The words *kado* (*corner*) and *kaado* (*card*) are spectrally identical, differing only in that *kado* is much shorter

in all contexts. The existence of such minimally-contrasting pairs is taken as conclusive evidence that *length* is phonemically distinctive for Japanese. As noted above, what is linguistically distinctive in any one language is generally present as a less *meaningful* signaling dimension in other languages. Thus, vowel length can be manipulated in any English word as well, but this occurs either consciously for emphasis or humorous effect, or unconsciously and very predictably at clause and sentence end positions, rather than to signal lexical identity in all contexts, as in Japanese.

Other interesting sounds that the English language makes no linguistic use of include the trilled *r* sound and the implosive. The trilled *r* sound is found in Spanish, distinguishing (for example) the words *pero* (*but*) and *perro* (*dog*). This trill could be found in times past as a non-lexical sound used for emphasis and interest by American circus ringmasters and other showpersons.

While the world's languages have all the variety of manner of articulation exemplified above and a great deal more, the primary dimension lacking in English that is exploited by a large subset of the world's languages is pitch variation. Many of the huge language families of Asia and Africa are tonal, including all varieties of Chinese. A large number of other languages are not considered strictly tonal by linguistics, yet they make systematic use of pitch contrasts. These include Japanese and Swedish. To be considered tonal, a language should have lexical meaning contrasts cued by pitch, just as the lexical meaning contrast between *pig* and *big* is cued by a voicing distinction in English. For example, Mandarin Chinese has four primary tones (tones can have minor context-dependent variants just like ordinary phones, as well) as shown in Table 2.11.

Table 2.11 The contrastive tones of Mandarin Chinese.

Tone	Shape	Example	Chinese	Meaning
1	High level	*ma*	妈	mother
2	High rising	*ma*	麻	numb
3	Low rising	*ma*	马	horse
4	High falling	*ma*	骂	to scold

Though English does not make systematic use of pitch in its inventory of word contrasts, nevertheless, as we always see with any possible phonetic effect, pitch is systematically varied in English to signal a speaker's emotions, intentions, and attitudes, and it has some linguistic function in signaling grammatical structure as well. Pitch variation in English will be considered in more detail in Chapter 15.

2.2.2. The Allophone: Sound and Context

The vowel and consonant charts provide abstract symbols for the phonemes – major sound distinctions. Phonemic units should be correlated with potential meaning distinctions. For example, the change created by holding the tongue high and front (*/iy/*) vs. directly down

from the (frontal) position for */eh/*, in the consonant context */m _ n/*, corresponds to an important meaning distinction in the lexicon of English: *mean /m iy n/* vs. *men /m eh n/*. This meaning contrast, conditioned by a pair of rather similar sounds, in an identical context, justifies the inclusion of */iy/* and */eh/* as logically separate distinctions.

However, one of the fundamental, meaning-distinguishing sounds is often modified in some systematic way by its phonetic neighbors. The process by which neighboring sounds influence one another is called *coarticulation*. Sometimes, when the variations resulting from coarticulatory processes can be consciously perceived, the modified phonemes are called *allophones*. Allophonic differences are always *categorical*, that is, they can be understood and denoted by means of a small, bounded number of symbols or diacritics on the basic phoneme symbols.

As an experiment, say the word *like* to yourself. Feel the front of the tongue touching the alveolar ridge (cf. Figure 2.21) when realizing the initial phoneme */l/*. This is one allophone of */l/*, the so-called *light* or *clear /l/*. Now say *kill*. In this word, most English speakers will no longer feel the front part of the tongue touch the alveolar ridge. Rather, the */l/* is realized by stiffening the broad midsection of the tongue in the rear part of the mouth while the continuant airstream escapes laterally. This is another allophone of */l/*, conditioned by its syllable-final position, called the *dark /l/*. Predictable contextual effects on the realization of phones can be viewed as a nuisance for speech recognition, as will be discussed in Chapter 9. On the other hand, such variation, because it is systematic, could also serve as a cue to the syllable, word, and prosodic structure of speech.

Now experiment with the sound */p/* by holding a piece of tissue in front of your mouth while saying the word *pin* in a normal voice. Now repeat this experiment with *spin*. For most English speakers, the word *pin* produces a noticeable puff of air, called aspiration. But the same phoneme, */p/*, embedded in the consonant cluster */sp/* loses its aspiration (burst, see the lines bracketing the */p/* release in *pin* and *spin* in Figure 2.22), and because these two types of */p/* are in complementary distribution (completely determined by phonetic and syllabic context), the difference is considered allophonic.

Try to speak the word *bat* in a framing phrase *say bat again*. Now speak *say bad again*. Can you feel the length difference in the vowel */ae/*? A vowel before a voiced consonant, e.g., */d/*, seems typically longer than the same vowel before the unvoiced counterpart, in this case */t/*.

A sound phonemicized as */t/* or */d/*, that is, a stop made with the front part of the tongue, may be reduced to a quick tongue tap that has a different sound than either */t/* or */d/* in fuller contexts. This process is called flapping. It occurs when */t/* or */d/* closes a stressed vowel (coda position) followed by an unstressed vowel, as in: *bitter, batter, murder, quarter, humidity,* and can even occur across words as long as the preconditions are met, as in *you can say that again*. Sometimes the velar flap opens too soon (anticipation), giving a characteristically nasal quality to some pre-nasal vowels such as */ae/* in *ham* vs. *had*. We have a more detailed discussion on allophones in Chapter 9.

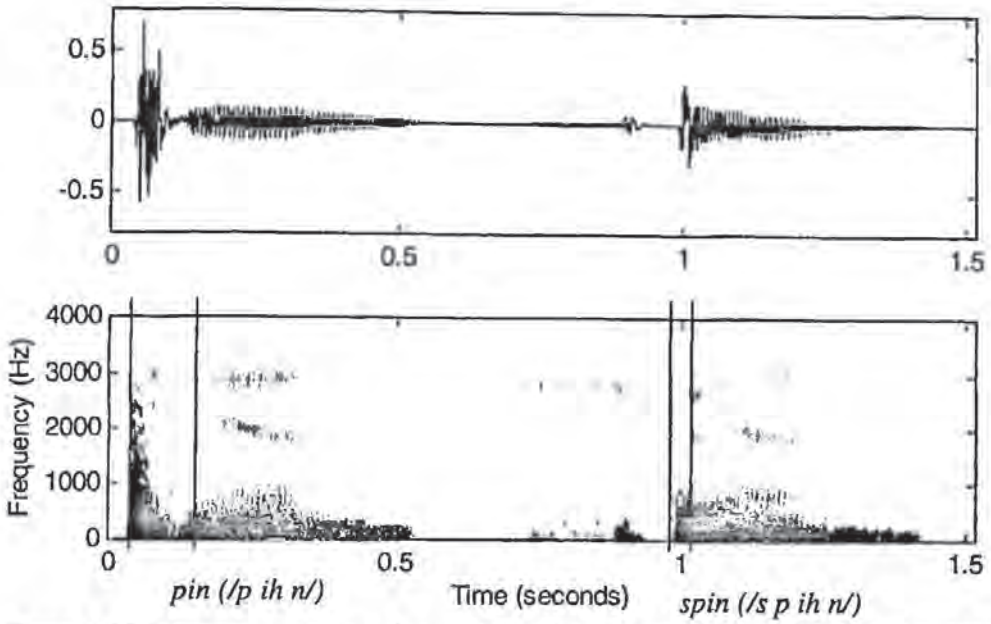


Figure 2.22 Spectrogram: bursts of *pin* and *spin*. The relative duration of a *p*-burst in different phonetic contexts is shown by the differing width of the area between the vertical lines.

2.2.3. Speech Rate and Coarticulation

In addition to allophones, there are other variations in speech for which no small set of established categories of variation can be established. These are *gradient*, existing along a scale for each relevant dimension, with speakers scattered widely. In general, it is harder to become consciously aware of coarticulation effects than of allophonic alternatives.

Individual speakers may vary their rates according to the content and setting of their speech, and there may be great inter-speaker differences as well. Some speakers may pause between every word, while others may speak hundreds of words per minute with barely a pause between sentences. At the faster rates, formant targets are less likely to be fully achieved. In addition, individual allophones may merge.

For example [20], consider the utterance *Did you hit it to Tom*? The pronunciation of this utterance is */d ih d y uw h ih t ih t t uw t aa m/*. However, a realistic, casual rendition of this sentence would appear as */d ih jh ax hh ih dx ih t ix t aa m/*, where */ix/* is a reduced schwa */ax/* that is short and often unvoiced, and */dx/* is a kind of shortened, indistinct stop, intermediate between */d/* and */t/*. The following five phonologic rules have operated on altering the pronunciation in the example:

- Palatalization of */d/* before */y/* in *did you*
- Reduction of unstressed */u/* to schwa in *you*

- Flapping of intervocalic /t/ in *hit it*
- Reduction of schwa and devoicing of /u/ in *to*
- Reduction of geminate (double consonant) /t/ in *it to*

There are also coarticulatory influences in the spectral appearance of speech sounds, which can only be understood at the level of spectral analysis. For example, in vowels, consonant neighbors can have a big effect on formant trajectories near the boundary. Consider the differences in *F1* and *F2* in the vowel */eh/* as realized in words with different initial consonants *bet*, *debt*, and *get*, corresponding to the three major places of articulation (labial, alveolar, and velar), illustrated in Figure 2.23. You can see the different relative spreads of *F1* and *F2* following the initial stop consonants.

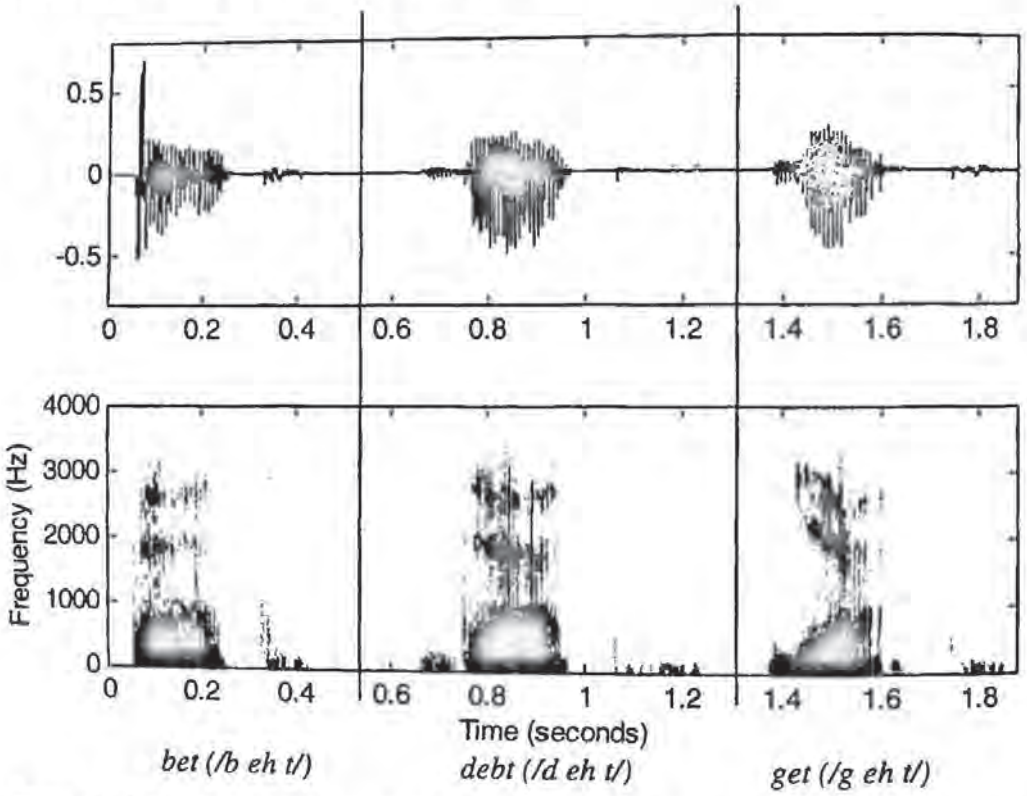


Figure 2.23 Spectrogram: *bet*, *debt*, and *get* (separated by vertical lines). Note different relative spreads of *F1* and *F2* following the initial stop consonants in each word.

Now let's see different consonants following the same vowel, *ebb*, *head*, and *egg*. In Figure 2.23, the coarticulatory effect is *perseverance*; i.e., in the early part of the vowel the articulators are still somewhat set from realization of the initial consonant. In the *ebb*, *head*, and *egg* examples shown in Figure 2.24, the coarticulatory effect is *anticipation*; i.e., in the latter part of the vowel the articulators are moving to prepare for the upcoming consonant articulation. You can see the increasing relative spread of *F1* and *F2* at the final vowel-consonant transition in each word.

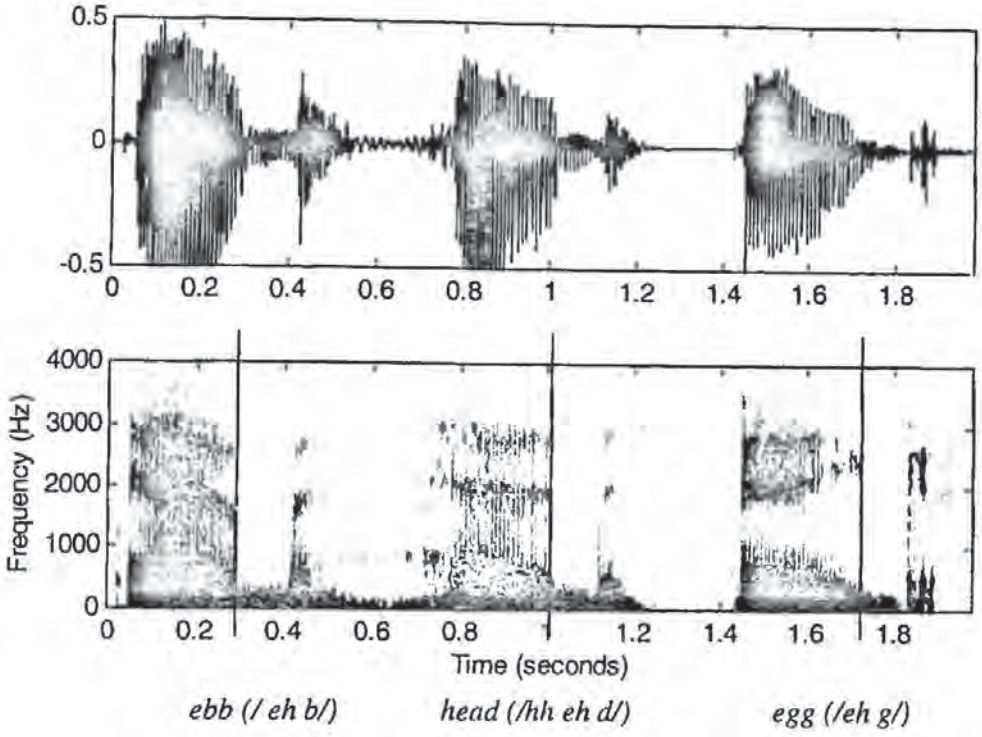


Figure 2.24 Spectrogram: *ebb*, *head*, and *egg*. Note the increasing relative spread of *F1* and *F2* at the final vowel-consonant transition in each word.

2.3. SYLLABLES AND WORDS

Phonemes are small building blocks. To contribute to language meaning, they must be organized into longer cohesive spans, and the units so formed must be combined in characteristic patterns to be meaningful, such as syllables and words in the English language.

2.3.1. Syllables

An intermediate unit, the *syllable*, is sometimes thought to interpose between the phones and the word level. The syllable is a slippery concept, with implications for both production and perception. Here we will treat it as a perceptual unit. Syllables are generally centered around vowels in English, giving two perceived syllables in a word like *tomcat*: */tOm-cAt/*. To completely parse a word into syllables requires making judgments of consonant affiliation (with the syllable peak vowels). The question of whether such judgments should be based on articulatory or perceptual criteria, and how they can be rigorously applied, remains unresolved.

Syllable centers can be thought of as *peaks* in sonority (high-amplitude, periodic sections of the speech waveform). These sonority peaks have affiliated *shoulders* of strictly non-increasing sonority. A scale of sonority can be used, ranking consonants along a continuum of stops, affricates, fricatives, and approximants. So, in a word like *verbal*, the syllabification would be *ver-bal*, or *verb-al*, but not *ve-rbal*, because putting the approximant /r/ before the stop /b/ in the second syllable would violate the non-decreasing sonority requirement heading into the syllable.

As long as the sonority conditions are met, the exact affiliation of a given consonant that could theoretically affiliate on either side can be ambiguous, unless determined by higher-order considerations of word structure, which may block affiliation. For example, in a word like *beekeeper*, an abstract boundary in the compound between the component words *bee* and *keeper* keeps us from accepting the syllable parse: *beek-eeper*, based on lexical interpretation. However, the same phonetic sequence in *beaker* could, depending on one's theory of syllabicity, permit affiliation of the *k: beak-er*. In general, the syllable is a unit that has intuitive plausibility but remains difficult to pin down precisely.

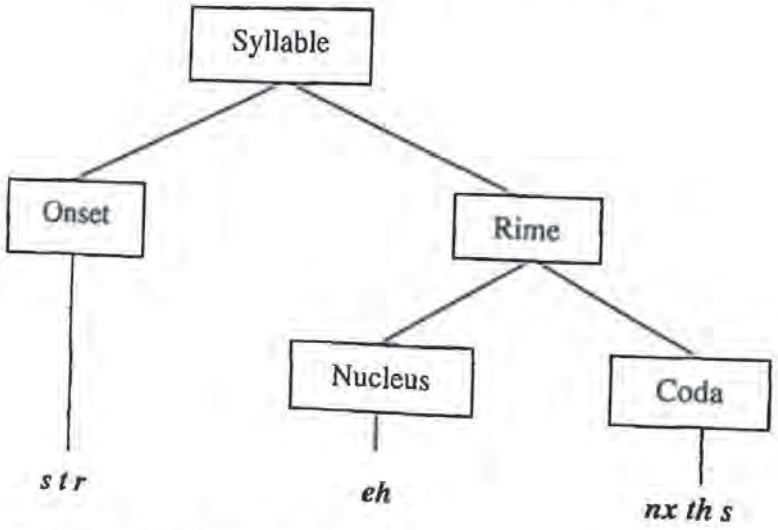


Figure 2.25 The word/syllable *strengths* (*/s t r eh nx th s/*) is the longest syllable of English.

Syllables are thought (by linguistic theorists) to have internal structure, and the terms used are worth knowing. Consider a big syllable such as *strengths /s t r eh nx th s/*. This consists of a vowel peak, called the *nucleus*, surrounded by the other sounds in characteristic positions. The *onset* consists of initial consonants if any, and the rime is the nucleus with trailing consonants (the part of the syllable that matters in determining poetic rhyme). The coda consists of consonants in the rime following the nucleus (in some treatments, the last consonant in a final cluster would belong to an *appendix*). This can be diagrammed as a syllable parse tree as shown in Figure 2.25. The syllable is sometimes thought to be the primary domain of coarticulation, that is, sounds within a syllable influence one another's realization more than the same sounds separated by a syllable boundary.

2.3.2. Words

The concept of words seems intuitively obvious to most speakers of Indo-European languages. It can be loosely defined as a lexical item, with an agreed-upon meaning in a given speech community, that has the freedom of syntactic combination allowed by its type (noun, verb, etc.).

In spoken language, there is a segmentation problem: words *run together* unless affected by a disfluency (unintended speech production problem) or by the deliberate placement of a pause (silence) for some structural or communicative reason. This is surprising to many people, because literacy has conditioned speakers/readers of Indo-European languages to expect a *blank space* between words on the printed page. But in speech, only a few true pauses (the aural equivalent of a blank space) may be present. So, what appears to the reading eye as *never give all the heart, for love* would appear to the ear, if we simply use letters to stand for their corresponding English speech sounds, as *nevergivealltheheart forlove* or, in phonemes, as *n eh v er g ih v ah l dh ax h aa r t \\ f ao r l ah v*. The \\ symbol marks a linguistically motivated pause, and the units so formed are sometimes called *intonation phrases*, as explained in Chapter 15.

Certain facts about word structure and combinatorial possibilities are evident to most native speakers and have been confirmed by decades of linguistic research. Some of these facts describe relations among words when considered in isolation, or concern groups of related words that seem intuitively similar along some dimension of form or meaning – these properties are *paradigmatic*. Paradigmatic properties of words include part-of-speech, inflectional and derivational morphology, and compound structure. Other properties of words concern their behavior and distribution when combined for communicative purposes in fully functioning utterances – these properties are *syntagmatic*.

2.3.2.1. Lexical Part-of-Speech

Lexical part-of-speech (POS) is a primitive form of linguistic theory that posits a restricted inventory of word-type categories, which capture generalizations of word forms and distributions. Assignment of a given POS specification to a word is a way of summarizing certain facts about its potential for syntagmatic combination. Additionally, paradigms of word formation processes are often similar within POS types and subtypes as well. The word properties upon which POS category assignments are based may include affixation behavior, very abstract semantic typologies, distributional patterns, compounding behavior, historical development, productivity and generalizabilty, and others.

A typical set of POS categories would include *noun*, *verb*, *adjective*, *adverb*, *interjection*, *conjunction*, *determiner*, *preposition*, and *pronoun*. Of these, we can observe that certain classes of words consist of infinitely large membership. This means new members can be added at any time. For example, the category of noun is constantly expanded to accommodate new inventions, such as Velcro or Spandex. New individuals are constantly being born, and their names are a type of noun called *proper noun*. The proliferation of words us-

ing the descriptive prefix *cyber* is another recent set of examples: *cyberscofflaw, cybersex,* and even *cyberia* illustrate the infinite creativity of humans in manipulating word structure to express new shades of meaning, frequently by analogy with, and using fragments of, existing vocabulary. Another example is the neologism *sheeple*, a noun combining the forms and meanings of *sheep* and *people* to refer to large masses of people who lack the capacity or willingness to take independent action. We can create new words whenever we like, but they had best fall within the predictable paradigmatic and syntagmatic patterns of use summarized by the existing POS generalizations, or there will be little hope of their adoption by any other speaker. These open POS categories are listed in Table 2.12. Nouns are inherently referential. They refer to persons, places, and things. Verbs are predicative; they indicate relations between entities and properties of entities, including participation in events. Adjectives typically describe and more completely specify noun reference, while adverbs describe, intensify, and more completely specify verbal relations. Open-class words are sometimes called *content* words, for their referential properties.

Table 2.12 Open POS categories.

Tag	Description	Function	Example
N	Noun	Names entity	*cat*
V	Verb	Names event or condition	*forget*
Adj	Adjective	Descriptive	*yellow*
Adv	Adverb	Manner of action	*quickly*
Interj	Interjection	Reaction	*oh!*

In contrast to the open-class categories, certain other categories of words only rarely and very slowly admit new members over the history of English development. These closed POS categories are shown in Table 2.13. The closed-category words are fairly stable over time. Conjunctions are used to join larger syntactically complete phrases. Determiners help to narrow noun reference possibilities. Prepositions denote common spatial and temporal relations of objects and actions to one another. Pronouns provide a convenient substitute for noun phrases that are fully understood from context. These words denote grammatical relations of other words to one another and fundamental properties of the world and how humans understand it. They can, of course, change slowly; for example, the Middle English pronoun *thee* is no longer in common use. The closed-class words are sometimes called *function* words.

Table 2.13 Closed POS categories.

Tag	Description	Function	Example
Conj	Conjunction	Coordinates phrases	*and*
Det	Determiner	Indicates definiteness	*the*
Prep	Preposition	Relations of time, space, direction	*from*
Pron	Pronoun	Simplified reference	*she*

The set of POS categories can be extended indefinitely. Examples can be drawn from the Penn Treebank project (http://www.cis.upenn.edu/ldc) as shown in Table 2.14, where you can find the proliferation of sub-categories, such as *Verb, base form* and *Verb, past tense*. These categories incorporate *morphological* attributes of words into the POS label system discussed in Section 2.3.2.2.

Table 2.14 Treebank POS categories – an expanded inventory.

String	Description	Example
CC	Coordinating conjunction	and
CD	Cardinal number	two
DT	Determiner	the
EX	Existential *there*	there (*There was an old lady*)
FW	Foreign word	omerta
IN	Preposition, subord. conjunction	over, but
JJ	Adjective	yellow
JJR	Adjective, comparative	better
JJS	Adjective, superlative	best
LS	List item marker	
MD	Modal	might
NN	Noun, singular or mass	rock, water
NNS	Noun, plural	rocks
NNP	Proper noun, singular	Joe
NNPS	Proper noun, plural	Red Guards
PDT	Predeterminer	all (*all the girls*)
POS	Possessive ending	's
PRP	Personal pronoun	I
PRP$	Possessive pronoun	mine
RB	Adverb	quickly
RBR	Adverb, comparative	higher (*shares closed higher.*)
RBS	Adverb, superlative	highest (*he jumped highest of all.*)
RP	Particle	up (*take up the cause*)
TO	*to*	to
UH	Interjection	hey!
VB	Verb, base form	choose
VBD	Verb, past tense	chose
VBG	Verb, gerund, or present participle	choosing
VBN	Verb, past participle	chosen
VBP	Verb, non-third person sing. present	jump
VBZ	Verb, third person singular present	jumps
WDT	Wh-determiner	which
WP	Wh-pronoun	who
WP$	Possessive wh-pronoun	whose
WRB	Wh-adverb	when (*When he came, it was late.*)

POS tagging is the process of assigning a part-of-speech or other lexical class marker to each word in a corpus. There are many algorithms to automatically tag input sentences into a set of tags. Rule-based methods [45], hidden Markov models (see Chapter 8) [23, 29, 46], and machine-learning methods [6] are used for this purpose.

2.3.2.2. Morphology

Morphology is about the subparts of words, i.e., the patterns of word formation including inflection, derivation, and the formation of compounds. English mainly uses prefixes and suffixes to express *inflection* and *derivational* morphology.

Inflectional morphology deals with variations in word form that reflect the contextual situation of a word in phrase or sentence syntax, and that rarely have direct effect on interpretation of the fundamental meaning expressed by the word. English inflectional morphology is relatively simple and includes person and number agreement and tense markings only. The variation in *cats* (vs. *cat*) is an example. The plural form is used to refer to an indefinite number of cats greater than one, depending on a particular situation. But the basic POS category (*noun*) and the basic meaning (*felis domesticus*) are not substantially affected. Words related to a common lemma via inflectional morphology are said to belong to a common paradigm, with a single POS category assignment. In English, common paradigm types include the verbal set of affixes (pieces of words): *-s, -ed, -ing*; the noun set: *-s*; and the adjectival *-er, -est*. Note that sometimes the base form may change spelling under affixation, complicating the job of automatic textual analysis methods. For historical reasons, certain paradigms may consist of highly idiosyncratic irregular variation as well, e.g., *go, going, went, gone* or *child, children*. Furthermore, some words may belong to defective paradigms, where only the singular (noun: *equipment*) or the plural (noun: *scissors*) is provided for.

In *derivational morphology*, a given root word may serve as the source for wholly new words, often with POS changes as illustrated in Table 2.15. For example, the terms *racial* and *racist*, though presumably based on a single root word *race*, have different POS possibilities (*adjective* vs. *noun-adjective*) and meanings. Derivational processes may induce pronunciation change or stress shift (e.g., *electric* vs. *electricity*). In English, typical derivational affixes (pieces of words) that are highly productive include prefixes and suffixes: *re-, pre-, -ial, -ism, -ish, -ity, -tion, -ness, -ment, -ious, -ify, -ize*, and others. In many cases, these can be added successively to create a complex layered form.

Table 2.15 Examples of stems and their related forms across POS categories.

Noun	Verb	Adjective	Adverb
criticism	*criticize*	*critical*	*critically*
fool	*fool*	*foolish*	*foolishly*
industry, industrialization	*industrialize*	*industrial,industrious*	*industriously*
employ, employee, employer	*employ*	*employable*	*employably*
certification	*certify*	*certifiable*	*certifiably*

Generally, word formation operates in layers, according to a kind of word syntax: *(deriv-prefix)* **root** (root)* (deriv-suffix)* (infl-suffix)*. This means that one or more *roots* can be compounded in the inner layer, with one or more optional *derivational prefixes*, followed by any number of optional *derivational suffixes*, capped off with no more than one *inflectional suffix*. There are, of course, limits on word formation, deriving both from semantics of the component words and simple lack of imagination. An example of a nearly maximal word in English might be *autocyberconceptualizations*, meaning (perhaps!) multiple instances of automatically creating computer-related concepts. This word lacks only compounding to be truly maximal. This word has a derivational prefix *auto-*, two root forms compounded (*cyber* and *concept*, though some may prefer to analyze *cyber-* as a prefix), three derivational suffixes (*-ual, -ize, -ation*), and is capped off with the plural inflectional suffix for nouns, *-s*.

2.3.2.3. Word Classes

POS classes are based on traditional grammatical and lexical analysis. With improved computational resources, it has become possible to examine words in context and assign words to groups according to their actual behavior in real text and speech from a statistical point of view. These kinds of classifications can be used in language modeling experiments for speech recognition, text analysis for text-to-speech synthesis, and other purposes.

One of the main advantages of word classification is its potential to derive more refined classes than traditional POS, while only rarely actually crossing traditional POS group boundaries. Such a system may group words automatically according to the similarity of usage with respect to their word neighbors. Consider classes automatically found by the classification algorithms of Brown *et al.* [7]:

{Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends}
{great big vast sudden mere sheer gigantic lifelong scant colossal}
{down backwards ashore sideways southward northward overboard aloft adrift}
{mother wife father son husband brother daughter sister boss uncle}
{John George James Bob Robert Paul William Jim David Mike}
{feet miles pounds degrees inches barrels tons acres meters bytes}

You can see that words are grouped together based on the semantic meaning, which is different from word classes created purely from syntactic point of view. Other types of classification are also possible, some of which can identify semantic relatedness across traditional POS categories. Some of the groups derived from this approach may include follows:

{problems problem solution solve analyzed solved solving}
{write writes writing written wrote pen}
{question questions asking answer answers answering}
{published publication author publish writer titled}

2.4. SYNTAX AND SEMANTICS

Syntax is the study of the patterns of formation of sentences and phrases from words and the rules for the formation of grammatical sentences. Semantics is another branch of linguistics dealing with the study of meaning, including the ways meaning is structured in language and changes in meaning and form over time.

2.4.1. Syntactic Constituents

Constituents represent the way a sentence can be divided into its grammatical subparts as constrained by common grammatical patterns (which implicitly incorporate normative judgments on acceptability). Syntactic constituents at least respect, and at best explain, the linear order of words in utterances and text. In this discussion, we will not strictly follow any of the many theories of syntax but will instead bring out a few basic ideas common to many approaches. We will not attempt anything like a complete presentation of the grammar of English but instead focus on a few simple phenomena.

Most work in syntactic theory has adopted machinery from traditional grammatical work on written language. Rather than analyze toy sentences, let's consider what kinds of superficial syntactic patterns are lurking in a random chunk of serious English text, excerpted from David Thoreau's essay *Civil Disobedience* [43]:

The authority of government, even such as I am willing to submit to – for I will cheerfully obey those who know and can do better than I, and in many things even those who neither know nor can do so well – is still an impure one: to be strictly just, it must have the sanction and consent of the governed. It can have no pure right over my person and property but what I concede to it. The progress from an absolute to a limited monarchy, from a limited monarchy to a democracy, is a progress toward a true respect for the individual.

2.4.1.1. Phrase Schemata

Words may be combined to form phrases that have internal structure and unity. We use generalized schemata to describe the phrase structure. The goal is to create a simple, uniform template that is independent of POS category.

Let's first consider nouns, a fundamental category referring to persons, places, and things in the world. The noun and its immediate modifiers form a constituent called the noun phrase (*NP*). To generalize this, we consider a word of arbitrary category, say category X (which could be a noun N or a verb V). The generalized rule for a phrase XP is $XP \Rightarrow$ *(modifiers) X-head (post-modifiers),* where X is the head, since it dominates the configuration and names the phrase. Elements preceding the head in its phrase are *premodifiers* and elements following the head are *postmodifiers*. XP, the culminating phrase node, is called a *maximal projection* of category X. We call the whole structure an *x-template*. Maximal projections, XP, are the primary currency of basic syntactic processes. The post-modifiers are usually maximal projections (another head, with its own post-modifiers forming an XP on its own) and are sometimes termed *complements*, because they are often required by the lexical properties of the head for a complete meaning to be expressed (e.g., when X is a preposition

or verb). Complements are typically noun phrases (*NP*), prepositional phrases (*PP*), verb phrases (*VP*), or sentence/clause (*S*), which make an essential contribution to the head's reference or meaning, and which the head requires for semantic completeness. Premodifiers are likely to be adverbs, adjectives, quantifiers, and determiners, i.e., words that help to specify the meaning of the head but may not be essential for completing the meaning. With minor variations, the *XP* template serves for most phrasal types, based on the POS of the head (*N, V, ADJ*, etc.).

For *NP*, we thus have *NP* $\Rightarrow$ *(det) (modifier)* ***head-noun*** *(post-modifier)*. This rule describes an *NP* (noun phrase – left side of arrow) in terms of its optional and required internal contents (right side of the arrow). *Det* is a word like *the* or *a* that helps to resolve the reference to a specific or an unknown instance of the noun. The *modifier* gives further information about the noun. The *head* of the phrase, and the only mandatory element, is the noun itself. *Post-modifiers* also give further information, usually in a more elaborate syntactic form than the simpler pre-modifiers, such as a relative clause or a prepositional phrase (covered below). The noun phrases of the passage above can be parsed as shown in Table 2.16. The head nouns may be personal pronouns (*I, it*), demonstrative and relative pronouns (*those*), coordinated nouns (*sanction and consent*), or common nouns (*individual*). The modifiers are mostly adjectives (*impure, pure*) or verbal forms functioning as adjectives (*limited*). The post-modifiers are interesting, in that, unlike the (pre-)modifiers, they are typically full phrases themselves, rather than isolated words. They include relative clauses (which are a kind of dependent sentence, e.g., *[those] who know and can do better than I*), as well as prepositional phrases (*of the governed*).

Table 2.16 *NPs* of the sample passage.

NP	Det	Mod	Head Noun	Post-Mod
1	the		**authority**	of government
2		even	**such**	as I am willing to submit to
3			**I**	
4			**those**	who know and can do better than I
5		many	**things**	
6		even	**those**	who neither know nor can do so well
7	an	impure	**one**	
8			**it**	
9	the		**sanction and consent**	of the governed
10	no	pure	**right**	over my person ... concede to it.
11	the		**progress**	from an absolute to a limited monarchy
12	an	absolute	**[monarchy]**	
13	a	limited	**monarchy**	
14	a		**democracy**	
15	a		**progress**	
16	a	true	**respect**	for the individual
17	the		**individual**	

Table 2.17 *PPs* of the sample passage.

Head Prep	Complement (Postmodifier)
of	Government
as	I am willing to submit to
than	I
in	many things
of	the governed
over	my person and property
to	it
from	an absolute [monarchy]
to	a limited monarchy
to	a democracy
toward	a true respect [for the individual]
for	the individual

Prepositions express spatial and temporal relations, among others. These are also said to project according to the *X*-template, but usually lack a pre-modifier. Some examples from the sample passage are listed in Table 2.17. The complements of *PP* are generally *NPs*, which may be simple head nouns like *government*. However, other complement types, such as the verb phrase in *after discussing it with Jo*, are also possible.

For verb phrases, the postmodifier (or complement) of a head verb would typically be one or more *NP* (noun phrase) maximal projections, which might, for example, function as a direct object in a *VP* like *pet the cat*. The complement may or may not be optional, depending on characteristics of the head. We can now make some language-specific generalizations about English. Some verbs, such as *give*, may take more than one kind of complement. So an appropriate template for a *VP* maximal projection in English would appear abstractly as *VP* $\Rightarrow$ *(modifier) verb (modifier) (Complement1, Complement2 ComplementN).* Complements are usually regarded as maximal projections, such as *NP*, *ADJP*, etc., and are enumerated in the template above, to cover possible multi-object verbs, such as *give*, which take both direct and indirect objects. Certain types of adverbs (*really*, *quickly*, *smoothly*, etc.) could be considered fillers for the *VP* modifier slots (before and after the head). In the sample passage, we find the following verb phrases as shown in Table 2.18.

VP presents some interesting issues. First, notice the multi-word verb *submit to*. Multi-word verbs such as *look after* and *put up with* are common. We also observe a number of auxiliary elements clustering before the verb in sentences of the sample passage: *am willing to submit to*, *will cheerfully obey*, and *can do better*. Rather than considering these as simple modifiers of the verbal head, they can be taken to have *scope* over the *VP* as a whole, which implies they are outside the *VP*. Since they are outside the *VP*, we can assume them to be heads in their own right, of phrases which require a *VP* as their complement. These elements mainly express tense (time or duration of verbal action) and modality (likelihood or probability of verbal action). In a full sentence, the *VP* has explicit or implicit inflection (projected from its verbal head) and indicates the person, number, and other context-dependent

features of the verb in relation to its arguments. In English, the person (first, second, third) and number (singular, plural) attributes, collectively called agreement features, of subject and verb must match. For simplicity, we will lump all these considerations together as inflectional elements, and posit yet another phrase type, the Inflectional Phrase (*IP*): *IP* ⇒ *premodifier head VP-complement.*

Table 2.18 *VP*s of the sample passage.

Pre-mod	Verb Head	Post-mod	Complement
	submit to		[the authority of government]
cheerfully	**obey**		those who know and can do better than I
	is	still	an impure one
	be		strictly just
	have		the sanction
	have		no pure right
	concede		to it
	is		a progress

The premodifier slot (sometimes called the *specifier* position in linguistic theory) of an *IP* is often filled by the subject of the sentence (typically a noun or *NP*). Since the *IP* unites the subject of a sentence with a *VP*, *IP* can also be considered simply as the sentence category, often written as *S* in speech grammars.

2.4.1.2. Clauses and Sentences

The *subject* of a sentence is what the sentence is mainly about. A *clause* is any phrase with both a subject and a *VP* (*predicate* in traditional grammars) that has potentially independent interpretation – thus, for us, a clause is an *IP*, a kind of sentence. A phrase is a constituent lacking either subject, predicate, or both. We have reviewed a number of phrase types above. There are also various types of clauses and sentences.

Even though clauses are sentences from an internal point of view (having subject and predicate), they often function as simpler phrases or words would, e.g., as modifiers (adjective and adverbs) or nouns and noun phrases. Clauses may appear as post-modifiers for nouns (so-called *relative clauses*), basically a kind of adjective clause, sharing their subjects with the containing sentence. Some clauses function as *NP*s in their own right. One common clause type substitutes a *wh-word* like *who* or *what* for a direct object of a verb in the embedded clause, to create a *questioned noun phrase* or indirect question: (*I don't know who Jo saw.*). In these clauses, it appears to syntacticians that the *questioned* object of the verb [*VP* saw **who**] has been extracted or moved to a new surface position (following the main clause verb *know*). This is sometimes shown in the phrase-structure diagram by co-indexing an empty *ghost* or trace constituent at the original position of the question pronoun with the question-*NP* appearing at the surface site:

I don't know $[_{NPobj}\,[_{IP}\,[_{NPi}$ *who] Jo saw* $[_{NPi}\,\_\,]]]$
$[_{NPsubj}[_{IP}$ *Whoever wins the game]] is our hero.*

There are various characteristic types of sentences. Some typical types include:

- Declarative: *I gave her a book.*
- Yes-no question: *Did you give her a book?*
- Wh-question: *What did you give her?*
- Alternatives question: *Did you give her a book, a scarf, or a knife?*
- Tag question: *You gave it to her, didn't you?*
- Passive: *She was given a book.*
- Cleft: *It must have been a book that she got.*
- Exclamative: *Hasn't this been a great birthday!*
- Imperative: *Give me the book.*

2.4.1.3. Parse Tree Representations

Sentences can be diagrammed in parse trees to indicate phrase-internal structure and linear precedence and immediate dominance among phrases. A typical phrase-structure tree for part of an embedded sentence is illustrated in Figure 2.26.

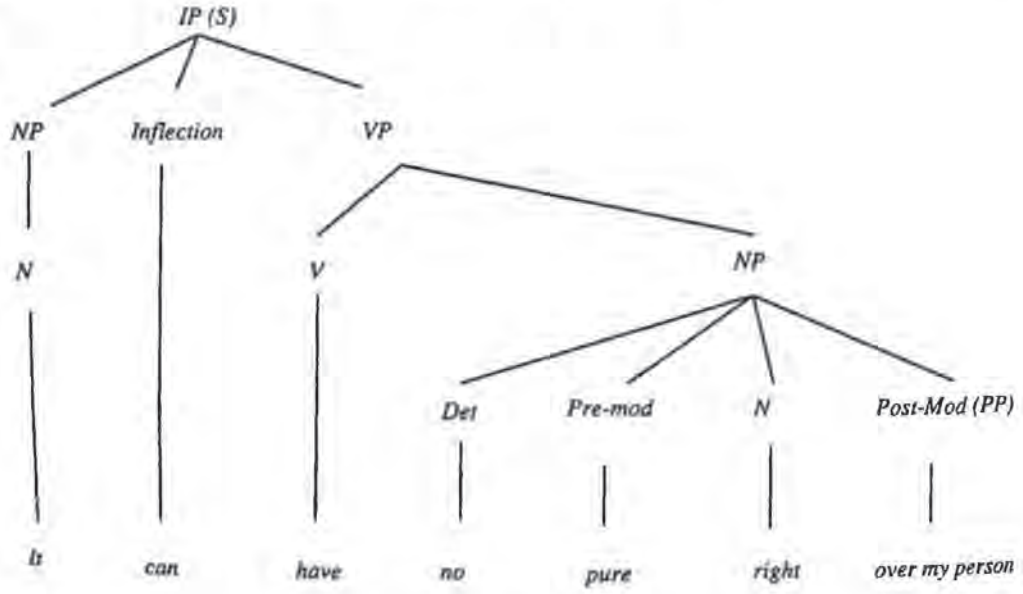


Figure 2.26 A simplified phrase-structure diagram.

For brevity, the same information illustrated in the tree can be represented as a bracketed string as follows:

$[_{IP}\,[_{NP}\,[_{N}\,It\,]_{N}]_{NP}\,[_{I}\,can\,]_{I}\,[_{VP}[_{V}\,have\,]_{V}\,[_{NP}\,no\ pure\ right\ [_{PP}\,over\ my\ person\,]_{PP}]_{NP}]_{VP}]_{IP}$

With such a bracketed representation, almost every type of syntactic constituent can be coordinated or joined with another of its type, and usually a new phrase node of the common type is added to subsume the constituents such as *NP: We have* $[_{NP}\,[_{NP}$ *tasty berries] and* $[_{NP}$ *tart juices]], IP/S:* $[_{IP}\,[_{IP}$ *Many have come] and* $[_{IP}$ *most have remained]], PP: We went* $[_{PP}\,[_{PP}$ *over the river] and* $[_{PP}$ *into the trees]],* and *VP: We want to* $[_{VP}\,[_{VP}$ *climb the mountains] and* $[_{VP}$ *sail the seas]].*

2.4.2. Semantic Roles

In traditional syntax, grammatical roles are used to describe the direction or control of action relative to the verb in a sentence. Examples include the ideas of *subject, object, indirect object*, etc. Semantic roles, sometimes called case relations, seem similar but dig deeper. They are used to make sense of the participants in an event, and they provide a vocabulary for us to answer the basic question *who did what to whom.* As developed by [13] and others, the theory of semantic roles posits a limited number of universal roles. Each basic meaning of each verb in our mental dictionary is tagged for the obligatory and optional semantic roles used to convey the particular meaning. A typical inventory of case roles is given below:

Agent	*cause or initiator of action, often intentional*
Patient/Theme	*undergoer of the action*
Instrument	*how action is accomplished*
Goal	*to whom action is directed*
Result	*result of action*
Location	*location of action*

These can be realized under various syntactic identities, and can be assigned to both required complement and optional adjuncts. A noun phrase in the Agentive role might be the surface subject of a sentence, or the object of the preposition *by* in a passive. For example, the verb *put* can be considered a process that has, in one of its senses, the case role specifications shown in Table 2.19.

Table 2.19 Analysis of a sentence with *put.*

Analysis	**Example**			
	Kim	*put*	*the book*	*on the table.*
Grammatical functions	*Subject (NP)*	*Predicate (VP)*	*Object (NP)*	*Adverbial (ADVP)*
Semantic roles	*Agent*	*Instrument*	*Theme*	*Location*

Now consider this passive-tense example, where the semantic roles align with different grammatical roles shown in Table 2.20. Words that look and sound identical can have different meaning or different *senses* as shown in Table 2.21. The sporting sense of *put* (as in the sport of shot-put) illustrates the meaning/sense-dependent nature of the role patterns, because in this sense the Locative case is no longer obligatory, as it is in the original sense illustrated in Table 2.19 and Table 2.20.

Table 2.20 Analysis of passive sentence with *put*.

Analysis	**Example**		
	The book	*was put*	*on the table.*
Grammatical functions	*Subject (NP)*	*Predicate (VP)*	*Adverbial (ADVP)*
Semantic roles	*Agent*	*Instrument*	*Location*

Table 2.21 Analysis of a different pattern of *put*.

Analysis	**Example**		
	Kim	*put*	*the shot.*
Grammatical functions	*Subject (NP)*	*Predicate (VP)*	*Object (NP)*
Semantic roles	*Agent*	*Instrument*	*Theme*

The lexical meaning of a verb can be further decomposed into primitive semantic relations such as CAUSE, CHANGE, and BE. The verb *open* might appear as *CAUSE(NP1,PHYSICAL-CHANGE(NP2,NOT-OPEN,OPEN))*. This says that for an agent (*NP1*) to *open* a theme (*NP2*) is to cause the patient to change from a not-opened state to an opened state. Such systems can be arbitrarily detailed and exhaustive, as the application requires.

2.4.3. Lexical Semantics

The specification of particular meaning templates for individual senses of particular words is called *lexical semantics*. When words combine, they may take on propositional meanings resulting from the composition of their meanings in isolation. We could imagine that a speaker starts with a proposition in mind (logical form as will be discussed in the next section), creating a need for particular words to express the idea (lexical semantics); the proposition is then linearized (syntactic form) and spoken (phonological/phonetic form). Lexical semantics is the level of meaning before words are composed into phrases and sentences, and it may heavily influence the possibilities for combination.

Words can be defined in a large number of ways including by relations to other words, in terms of decomposition semantic primitives, and in terms of non-linguistic cognitive constructs, such as perception, action, and emotion. There are hierarchical and non-hierarchical relations. The main hierarchical relations would be familiar to most object-oriented programmers. One is *is-a* taxonomies (a *crow* is-a *bird*), which have transitivity of properties

from type to subtype (inheritance). Another is *has-a* relations (a *car* has-a *windshield*), which are of several differing qualities, including process/subprocess (*teaching* has-a subprocess *giving exams*), and arbitrary or natural subdivisions of part-whole relations (*bread* has-a division into *slices*, *meter* has-a division into *centimeters*). Then there are non-branching hierarchies (no fancy name) that essentially form scales of degree, such as *frozen* $\Rightarrow$ *cold* $\Rightarrow$ *lukewarm* $\Rightarrow$ *hot* $\Rightarrow$ *burning*. Non-hierarchical relations include synonyms, such as *big/large*, and antonyms such as *good/bad*.

Words seem to have natural affinities and disaffinities in the semantic relations among the concepts they express. Because these affinities could potentially be exploited by future language understanding systems, researchers have used the generalizations above in an attempt to tease out a parsimonious and specific set of basic relations under which to group entire lexicons of words. A comprehensive listing of the families and subtypes of possible semantic relations has been presented in [10]. In Table 2.22, the leftmost column shows names for families of proposed relations, the middle column differentiates subtypes within each family, and the rightmost column provides examples of word pairs that participate in the proposed relation. Note that case roles have been modified for inclusion as a type of semantic relation within the lexicon.

We can see from Table 2.22 that a single word could participate in multiple relations of different kinds. For example, *knife* appears in the examples for *Similars: invited attribute* (i.e., a desired and expected property) as: *knife-sharp*, and also under *Case Relations*: *action-instrument*, which would label the relation of *knife* to the action *cut* in *He cut the bread with a knife*. This suggests that an entire lexicon could be viewed as a graph of semantic relations, with words or idioms as nodes and connecting edges between them representing semantic relations as listed above. There is a rich tradition of research in this vein.

The biggest practical problem of lexical semantics is the context-dependent resolution of senses of words – so-called polysemy. A classic example is *bank – bank of the stream* as opposed to *money in the bank*. While lexicographers try to identify distinct senses when they write dictionary entries, it has been generally difficult to rigorously quantify exactly what counts as a discrete sense of a word and to disambiguate the senses in practical contexts. Therefore, designers of practical speech understanding systems generally avoid the problem by limiting the domain of discourse. For example, in a financial application, generally only the sense of *bank* as a fiduciary institution is accessible, and others are assumed not to exist. It is sometimes difficult to make a principled argument as to how many distinct senses a word has, because at some level of depth and abstraction, what might appears as separate senses seem to be similar or related, as *face* could be *face of a clock* or *face of person*.

Senses are usually distinguished within a given part-of-speech (POS) category. Thus, when an occurrence of *bank* has been identified as a verb, the *shore* sense might be automatically eliminated, though depending on the sophistication of the system's lexicon and goals, there can be sense differences for many English verbs as well. Within a POS category, often the words that occur near a given ambiguous form in the utterance or discourse are clues to interpretation, where links can be established using semantic relations as described above. Mutual information measures as discussed in Chapter 3 can sometimes provide hints. In a context of dialog where other, less ambiguous financial terms come up

frequently, the sense of *bank* as fiduciary institution is more likely. Finally, when all else fails, often senses can be ranked in terms of their a priori likelihood of occurrence. It should always be borne in mind that language is not static; it can change form under a given analysis at any time. For example, the stable English form *spinster*, a somewhat pejorative term for an older, never-married female, has recently taken on a new morphologically complex form, with the new sense of a high political official, or media spokesperson, employed to provide bland disinformation (*spin*) on a given topic.

Table 2.22 Semantic relations.

Family	**Subtype**	**Example**
Contrasts	Contrary	*old-young*
	Contradictory	*alive-dead*
	Reverse	*buy-sell*
	Directional	*front-back*
	Incompatible	*happy-morbid*
	Asymmetric contrary	*hot-cool*
	Attribute similar	*rake-fork*
Similars	Synonymity	*car-auto*
	Dimensional similar	*smile-laugh*
	Necessary attribute	*bachelor-unmarried*
	Invited attribute	*knife-sharp*
	Action subordinate	*talk-lecture*
Class Inclusion	Perceptual subord.	*animal-horse*
	Functional subord.	*furniture-chair*
	State subord.	*disease-polio*
	Activity subord.	*game-chess*
	Geographic subord.	*country-Russia*
	Place	*Germany-Hamburg*
Case Relations	Agent-action	*artist-paint*
	Agent-instrument	*farmer-tractor*
	Agent-object	*baker-bread*
	Action-recipient	*sit-chair*
	Action-instrument	*cut-knife*
Part-Whole	Functional object	*engine-car*
	Collection	*forest-tree*
	Group	*choir-singer*
	Ingredient	*table-wood*
	Functional location	*kitchen-stove*
	Organization	*college-admissions*
	Measure	*mile-yard*

2.4.4. Logical Form

Because of all the lexical, syntactic, and semantic ambiguity in language, some of which requires external context for resolution, it is desirable to have a metalanguage in which to concretely and succinctly express all linguistically possible meanings of an utterance before discourse and world knowledge are applied to choose the most likely interpretation. The favored metalanguage for this purpose is called the predicate logic, used to represent the logical form, or context-independent meaning, of an utterance. The semantic component of many SLU architectures builds on a substrate of two-valued, first-order, logic. To distinguish *shades of meaning* beyond truth and falsity requires more powerful formalisms for knowledge representation.

In a typical first-order system, predicates correspond to events or conditions denoted by verbs (such as *Believe* or *Like*), states of identity (such as being a *Dog* or *Cat*), and properties of varying degrees of permanence (*Happy*). In this form of logical notation, predicates have open places, filled by arguments, as in a programming language subroutine definition. Since individuals may have identical names, subscripting can be used to preserve unique reference. In the simplest systems, predication ranges over individuals rather than higher-order entities such as properties and relations.

Predicates with filled argument slots map onto sets of individuals (constants) in the universe of discourse, in particular those individuals possessing the properties, or participating in the relation, named by the predicate. One-place predicates like *Soldier*, *Happy*, or *Sleeps* range over sets of individuals from the universe of discourse. Two-place predicates, like transitive verbs such as *loves*, range over a set consisting of ordered pairs of individual members (constants) of the universe of discourse. For example, we can consider the universe of discourse to be U = {*Romeo, Juliet, Paris, Rosaline, Tybalt*}, people as characters in a play. They do things with and to one another, such as loving and killing. Then we could imagine the relation *Loves* interpreted as the set of ordered pairs: {<*Romeo, Juliet*>, <*Juliet, Romeo*>, <*Tybalt, Tybalt*>, <*Paris, Juliet*>}, a subset of the Cartesian product of theoretically possible love matches $U \times U$. So, for any ordered pair x, y in U, *Loves*(x, y) is true if the ordered pair <x,y> is a member of the extension of the *Loves* predicate as defined, e.g., *Romeo loves Juliet, Juliet loves Romeo*, etc.. Typical formal properties of relations are sometimes specially marked by grammar, such as the reflexive relation *Loves*(*Tybalt, Tybalt*), which can rendered in natural language as *Tybalt loves himself*. Not every possibility is present; for instance in our example, the individual Rosaline does not happen to participate at all in this extensional definition of *Loves* over U, as her omission from the pairs list indicates. Notice that the subset of *Loves*(x, y) of ordered pairs involving both *Romeo* and *Juliet* is symmetric, also marked by grammar, as in *Romeo and Juliet love each other*. This general approach extends to predicates with any arbitrary number of arguments, such as intransitive verbs like *give*.

Just as in ordinary propositional logic, connectives such as negation, conjunction, disjunction, and entailment are admitted, and can be used with predicates to denote common natural language meanings:

Romeo isn't happy = ¬Happy(Romeo)
Romeo isn't happy, but Tybalt is (happy) = ¬Happy(Romeo) ∧ Happy(Tybalt)
Either Romeo or Tybalt is happy = Happy(Romeo) ∨ Happy(Tybalt)
If Romeo is happy, Juliet is happy = Happy(Romeo) → Happy(Juliet)

Formulae, such as those above, are also said to bear a binary truth value, true or false, with respect to a world of individuals and relations. The determination of the truth value is compositional, in the sense that the truth value of the whole depends on the truth value of the parts. This is a simplistic but formally tractable view of the relation between language and meaning.

Predicate logic can also be used to denote quantified noun phrases. Consider a simple case such as *Someone killed Tybalt*, predicated over our same U = {*Romeo, Juliet, Paris, Rosaline, Tybalt*}. We can now add an *existential* quantifier, ∃, standing for *there exists* or *there is at least one*. This quantifier will bind a variable over individuals in U, and will attach to a proposition to create a new, quantified proposition in logical form. The use of variables in propositions such as *killed(x, y)* creates open propositions. Binding the variables with a quantifier over them closes the proposition. The quantifier is prefixed to the original proposition: ∃*x Killed(x, Tybalt)*.

To establish a truth (semantic) value for the quantified proposition, we have to satisfy the disjunction of propositions in U: *Killed(Romeo, Tybalt)* ∨ *Killed(Juliet, Tybalt)* ∨ *Killed(Paris, Tybalt)* ∨ *Killed(Rosaline, Tybalt)* ∨ *Killed(Tybalt, Tybalt)*. The set of all such bindings of the variable x is the space that determines the truth or falsity of the proposition. In this case, the binding of x = *Romeo* is sufficient to assign a value true to the existential proposition.

2.5. Historical Perspective and Further Reading

Motivated to improve speech quality over the telephone, AT&T Bell Labs has contributed many influential discoveries in speech hearing, including the critical band and articulation index [2, 3]. The *Auditory Demonstration* CD prepared by Houtsma, Rossing, and Wagenaars [18] has a number of very interesting examples on psychoacoustics and its explanations. *Speech, Language, and Communication* [30] and *Speech Communication – Human and Machine* [32] are two good books that provide modern introductions to the structure of spoken language. Many speech perception experiments were conducted by exploring how phonetic information is distributed in the time or frequency domain. In addition to the formant structures for vowels, frequency importance function [12] has been developed to study how features related to phonetic categories are stored at various frequencies. In the time domain, it has been observed [16, 19, 42] that salient perceptual cues may not be evenly distributed over the speech segments and that certain perceptual critical points exist.

As intimate as speech and acoustic perception may be, there are also strong evidences that lexical and linguistic effects on speech perception are not always consistent with acoustic ones. For instance, it has long been observed that humans exhibit difficulties in distinguishing non-native phonemes. Human subjects also carry out categorical goodness

difference assimilation based on their mother tongue [34], and such perceptual mechanism can be observed as early as in six-month-old infants [22]. On the other hand, hearing-impaired listeners are able to effortlessly overcome their acoustical disabilities for speech perception [8]. Speech perception is not simply an auditory matter. McGurk and MacDonald (1976) [27, 28] dramatically demonstrated this when they created a videotape on which the auditory information (phonemes) did not match the visual speech information. The effect of this mismatch between the auditory signal and the visual signal was to create a third phoneme different from both the original auditory and visual speech signals. An example is dubbing the phoneme */ba/* to the visual speech movements */ga/*. This mismatch results in hearing the phoneme */da/*. Even when subjects know of the effect, they report the McGurk effect percept. The McGurk effect has been demonstrated for consonants, vowels, words, and sentences.

The earliest scientific work on phonology and grammars goes back to Panini, a Sanskrit grammarian of the fifth century B.C. (estimated), who created a comprehensive and scientific theory of phonetics, phonology, and morphology, based on data from Sanskrit (the classical literary language of the ancient Hindus). Panini created formal production rules and definitions to describe Sanskrit grammar, including phenomena such as construction of sentences, compound nouns, etc. Panini's formalisms function as ordered rules operating on underlying structures in a manner analogous to modern linguistic theory. Panini's phonological rules are equivalent in formal power to Backus-Nauer form (BNF). A general introduction to this pioneering scientist is Cardona [9].

An excellent introduction to all aspects of phonetics is *A Course in Phonetics* [24]. A good treatment of the acoustic structure of English speech sounds and a through introduction and comparison of theories of speech perception is to be found in [33]. The basics of phonology as part of linguistic theory are treated in *Understanding Phonology* [17]. An interesting treatment of word structure (morphology) from a computational point of view can be found in *Morphology and Computation* [40]. A comprehensive yet readable treatment of English syntax and grammar can be found in *English Syntax* [4] and *A Comprehensive Grammar of the English Language* [36]. Syntactic theory has traditionally been the heart of linguistics, and has been an exciting and controversial area of research since the 1950s. Be aware that almost any work in this area will adopt and promote a particular viewpoint, often to the exclusion or minimization of others. A reasonable place to begin with syntactic theory is *Syntax: A Minimalist Introduction* [37]. An introductory textbook on syntactic and semantic theory that smoothly introduces computational issues is *Syntactic Theory: A Formal Introduction* [39]. For a philosophical and entertaining overview of various aspects of linguistic theory, see *Rhyme and Reason: An Introduction to Minimalist Syntax* [44]. A good and fairly concise treatment of basic semantics is *Introduction to Natural Language Semantics* [11]. Deeper issues are covered in greater detail and at a more advanced level in *The Handbook of Contemporary Semantic Theory* [25]. The intriguing area of lexical semantics (theory of word meanings) is comprehensively presented in *The Generative Lexicon* [35]. *Concise History of the Language Sciences* [21] is a good edited book if you are interested in the history of linguistics.

REFERENCES

[1] Aliprand, J., *et al.*, *The Unicode Standard, Version 2.0*, 1996, Addison Wesley.

[2] Allen, J.B., "How Do Humans Process and Recognize Speech?," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(4), pp. 567-577.

[3] Allen, J.B., "Harvey Fletcher 1884–1981" in *The ASA Edition of Speech and Hearing Communication* 1995, Woodbury, New York, pp. A1-A34, Acoustical Society of America.

[4] Baker, C.L., *English Syntax*, 1995, Cambridge, MA, MIT Press.

[5] Blauert, J., *Spatial Hearing*, 1983, MIT Press.

[6] Brill, E., "Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging," *Computational Linguistics*, 1995, **21**(4), pp. 543-566.

[7] Brown, P., *et al.*, "Class-Based N-gram Models of Natural Language," *Computational Linguistics*, 1992, **18**(4).

[8] Caplan, D. and J. Utman, "Selective Acoustic Phonetic Impairment and Lexical Access in an Aphasic Patient," *Journal of the Acoustical Society of America*, 1994, **95**(1), pp. 512-517.

[9] Cardona, G., *Panini: His Work and Its Traditions: Background and Introduction*, 1988, Motilal Banarsidass.

[10] Chaffin, R., and Herrmann, D., "The Nature of Semantic Relations: A Comparison of Two Approaches" in *Representing Knowledge in Semantic Networks*, M. Evens, ed., 1988, Cambridge, UK, Cambridge University Press.

[11] de Swart, H., *Introduction to Natural Language Semantics*, 1998, Stanford, CA, Center for the Study of Language and Information Publications.

[12] Duggirala, V., *et al.*, "Frequency Importance Function for a Feature Recognition Test Material," *Journal of the Acoustical Society of America*, 1988, **83**(9), pp. 2372-2382.

[13] Fillmore, C.J., "The Case for Case" in *Universals in Linguistic Theory*, E. Bach and R. Harms, eds. 1968, New York, NY, Holt, Rinehart and Winston.

[14] Fletcher, H., "Auditory Patterns," *Rev. Mod. Phys.*, 1940, **12**, pp. 47-65.

[15] Fry, D.B., *The Physics of Speech*, Cambridge Textbooks in Linguistics, 1979, Cambridge, U.K., Cambridge University Press.

[16] Furui, S., "On The Role of Spectral Transition for Speech Perception," *Journal of the Acoustical Society of America*, 1986, **80**(4), pp. 1016-1025.

[17] Gussenhoven, C., and Jacobs, H., *Understanding Phonology*, Understanding Language Series, 1998, Edward Arnold.

[18] Houtsma, A., T. Rossing, and W. Wagenaars, *Auditory Demonstrations*, 1987, Institute for Perception Research, Eindhovern, The Netherlands, Acoustic Society of America.

[19] Jenkins, J., W. Strange, and S. Miranda, "Vowel Identification in Mixed-Speaker Silent-Center Syllables," *Journal of the Acoustical Society of America*, 1994, **95**(2), pp. 1030-1041.

[20] Klatt, D., "Review of the ARPA Speech Understanding Project," *Journal of Acoustical Society of America*, 1977, **62**(6), pp. 1324-1366.

[21] Koerner, E. and E. Asher, eds. *Concise History of the Language Sciences*, 1995, Oxford, Elsevier Science.

[22] Kuhl, P., "Infant's Perception and Representation of Speech: Development of a New Theory," *Int. Conf. on Spoken Language Processing*, 1992, Alberta, Canada, pp. 449-452.

[23] Kupeic, J., "Robust Part-of-Speech Tagging Using a Hidden Markov Model," *Computer Speech and Language*, 1992, **6**, pp. 225-242.

[24] Ladefoged, P., *A Course in Phonetics*, 1993, Harcourt Brace Johanovich.

[25] Lappin, S., *The Handbook of Contemporary Semantic Theory*, Blackwell Handbooks in Linguistics, 1997, Oxford, UK, Blackwell Publishsers Inc.

[26] Lindsey, P. and D. Norman, *Human Information Processing*, 1972, New York and London, Academic Press.

[27] MacDonald, J. and H. McGurk, "Visual Influence on Speech Perception Process," *Perception and Psychophysics*, 1978, **24**(3), pp. 253-257.

[28] McGurk, H. and J. MacDonald, "Hearing Lips and Seeing Voices," *Nature*, 1976, **264**, pp. 746-748.

[29] Merialdo, B., "Tagging English Text with a Probabilistic Model," *Computational Linguistics*, 1994, **20**(2), pp. 155-172.

[30] Miller, J. and P. Eimas, *Speech, Language and Communication*, Handbook of Perception and Cognition, eds. E. Carterette and M. Friedman, 1995, Academic Press.

[31] Moore, B.C., *An Introduction to the Psychology of Hearing*, 1982, London, Academic Press.

[32] O'Shaughnessy, D., *Speech Communication – Human and Machine*, 1987, Addison-Wesley.

[33] Pickett, J.M., *The Acoustics of Speech Communication*, 1999, Needham Heights, MA, Allyn & Bacon.

[34] Polka, L., "Linguistic Influences in Adult Perception of Non-native Vowel Contrast," *Journal of the Acoustical Society of America*, 1995, **97**(2), pp. 1286-1296.

[35] Pustejovsky, J., *The Generative Lexicon*, 1998, Bradford Books.

[36] Quirk, R., Svartvik, J., Leech, G., *A Comprehensive Grammar of the English Language*, 1985, Addison-Wesley Pub. Co.

[37] Radford, A., *Syntax: A Minimalist Introduction*, 1997, Cambridge, U.K., Cambridge Univ. Press.

[38] Rossing, T.D., *The Science of Sound*, 1982, Reading, MA, Addison-Wesley.

[39] Sag, I., Wasow, T., *Syntactic Theory: A Formal Introduction*, 1999, Cambridge, UK, Cambridge University Press.

[40] Sproat, R., *Morphology and Computation*, ACL-MIT Press Series in Natural Language Processing, 1992, Cambridge, MA, MIT Press.

[41] Stevens, S.S. and J. Volkman, "The Relation of Pitch to Frequency," *Journal of Psychology*, 1940, **53**, pp. 329.

[42] Strange, W., J. Jenkins, and T. Johnson, "Dynamic Specification of Coarticulated Vowels," *Journal of the Acoustical Society of America*, 1983, 74(3), pp. 695-705.

[43] Thoreau, H.D., *Civil Disobedience, Solitude and Life Without Principle*, 1998, Prometheus Books.

[44] Uriagereka, J., *Rhyme and Reason: An Introduction to Minimalist Syntax*, 1998, Cambridge, MA, MIT Press.

[45] Voutilainen, A., "Morphological Disambiguation" in *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text* 1995, Berlin, Mouton de Gruyter.

[46] Weischedel, R., "BBN: Description of the PLUM System as Used for MUC-6," *The 6th Message Understanding Conferences (MUC-6)*, 1995, San Francisco, Morgan Kaufmann, pp. 55-70.

CHAPTER 3

Probability, Statistics, and Information Theory

Randomness and uncertainty play an important role in science and engineering. Most spoken language processing problems can be characterized in a probabilistic framework. Probability theory and statistics provide the mathematical language to describe and analyze such systems.

The criteria and methods used to estimate the unknown probabilities and probability densities form the basis for estimation theory. Estimation theory is critical to parameter learning in pattern recognition. In this chapter, three widely used estimation methods are discussed. They are *minimum mean squared error estimation* (MMSE), *maximum likelihood estimation* (MLE), and *maximum posterior probability estimation* (MAP).

Significance testing deals with the confidence of statistical inference, such as knowing whether the estimation of some parameter can be accepted with confidence. In pattern recognition, significance testing is important for determining whether the observed difference between two different classifiers is real. In our coverage of significance testing, we describe various methods that are used in pattern recognition, discussed in Chapter 4.

Information theory was originally developed for efficient and reliable communication systems. It has evolved into a mathematical theory concerned with the very essence of the communication process. It provides a framework for the study of fundamental issues, such as the efficiency of information representation and the limitations in reliable transmission of information over a communication channel. Many of these problems are fundamental to spoken language processing.

3.1. PROBABILITY THEORY

Probability theory deals with the averages of mass phenomena occurring sequentially or simultaneously. We often use probabilistic expressions in our day-to-day lives, such as when saying, *It is very likely that the Dow (Dow Jones Industrial index) will hit 12,000 points next month*, or, *The chance of scattered showers in Seattle this weekend is high.* Each of these expressions is based upon the concept of the probability, or the likelihood, that some specific event will occur.

Probability can be used to represent the degree of confidence in the outcome of some actions (observations), which are not definite. In probability theory, the term *sample space, S,* is used to refer to the collection (set) of all possible outcomes. An *event* refers to a subset of the sample space or a collection of outcomes. The *probability of event A,* denoted as $P(A)$, can be interpreted as the *relative frequency* with which event A would occur if the process were repeated a large number of times under similar conditions. Based on this interpretation, $P(A)$ can be computed simply by counting the total number, N_S, of all observations and the number of observations N_A whose outcome belongs to event A. That is,

$$P(A) = \frac{N_A}{N_S} \tag{3.1}$$

$P(A)$ is bounded between zero and one, i.e.,

$$0 \le P(A) \le 1 \text{ for all } A \tag{3.2}$$

The lower bound of probability $P(A)$ is zero when the event set A is an empty set. On the other hand, the upper bound of probability $P(A)$ is one when the event set A happens to be S.

If there are n events $A_1, A_2, \cdots, A_n$ in S such that $A_1, A_2, \cdots, A_n$ are disjoint and $\bigcup_{i=1}^{n} A_i = S$, events $A_1, A_2, \cdots, A_n$ are said to form a *partition* of S. The following obvious equation forms a fundamental axiom for probability theory.

$$P(A_1 \cup A_2 \cup \ldots A_n) = \sum_{i=1}^{n} P(A_i) = 1 \tag{3.3}$$

Based on the definition in Eq. (3.1), the *joint probability* of event A and event B occurring concurrently is denoted as $P(AB)$ and can be calculated as:

$$P(AB) = \frac{N_{AB}}{N_S} \tag{3.4}$$

3.1.1. Conditional Probability and Bayes' Rule

It is useful to study the way in which the probability of an event A changes after it has been learned that some other event B has occurred. This new probability denoted as $P(A \mid B)$ is called the *conditional probability* of event A given that event B has occurred. Since the set of those outcomes in B that also result in the occurrence of A is exactly the set AB as illustrated in Figure 3.1, it is natural to define the conditional probability as the proportion of the total probability $P(B)$ that is represented by the joint probability $P(AB)$. This leads to the following definition:

$$P(A \mid B) = \frac{P(AB)}{P(B)} = \frac{N_{AB}/N_S}{N_B/N_S} \tag{3.5}$$

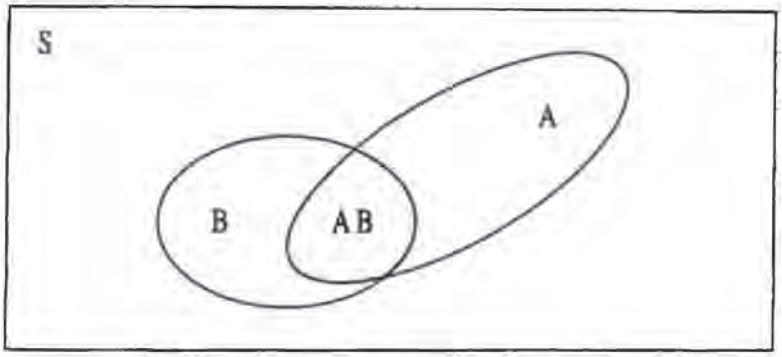


Figure 3.1 The intersection AB represents where the joint event A and B occurs concurrently.

Based on the definition of conditional probability, the following expressions can be easily derived.

$$P(AB) = P(A \mid B)P(B) = P(B \mid A)P(A) \tag{3.6}$$

Equation (3.6) is the simple version of the *chain rule*. The chain rule, which can specify a joint probability in terms of multiplication of several cascaded conditional probabilities, is often used to decompose a complicated joint probabilistic problem into a sequence of stepwise conditional probabilistic problems. Equation (3.6) can be converted to such a general chain:

$$P(A_1A_2 \cdots A_n) = P(A_n \mid A_1 \cdots A_{n-1}) \cdots P(A_2 \mid A_1)P(A_1) \tag{3.7}$$

When two events, A and B, are independent of each other, in the sense that the occurrence or of either of them has no relation to and no influence on the occurrence of the other, it is obvious that the conditional probability $P(B \mid A)$ equals to the unconditional probability

$P(B)$. It follows that the joint probability $P(AB)$ is simply the product of $P(A)$ and $P(B)$ if A and B, are independent.

If the n events $A_1, A_2, \cdots, A_n$ form a partition of S and B is any event in S as illustrated in Figure 3.2, the events $A_1B, A_2B, \cdots, A_nB$ form a partition of B. Thus, we can rewrite:

$$B = A_1B \cup A_2B \cup \cdots \cup A_nB \tag{3.8}$$

Since $A_1B, A_2B, \cdots, A_nB$ are disjoint,

$$P(B) = \sum_{k=1}^{n} P(A_kB) \tag{3.9}$$

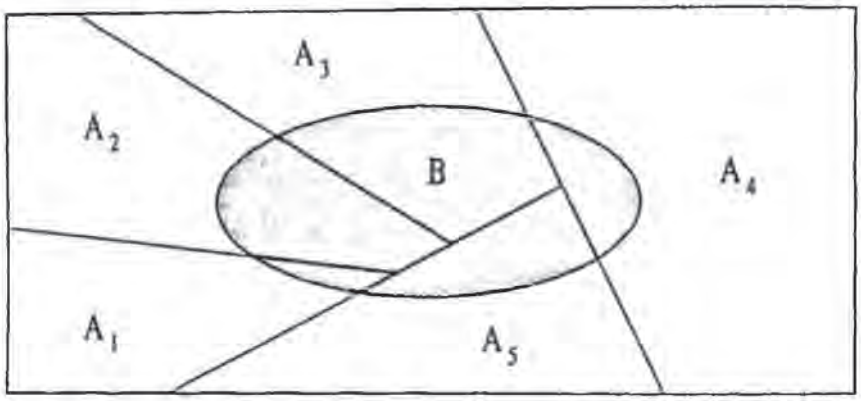


Figure 3.2 The intersections of B with partition events $A_1, A_2, \cdots, A_n$.

Equation (3.9) is called the *marginal probability* of event B, where the probability of event B is computed from the sum of joint probabilities.

According to the chain rule, Eq. (3.6), $P(A_iB) = P(A_i)P(B \mid A_i)$, it follows that

$$P(B) = \sum_{k=1}^{n} P(A_k)P(B \mid A_k) \tag{3.10}$$

Combining Eqs. (3.5) and (3.10), we get the well-known *Bayes' rule*:

$$P(A_i \mid B) = \frac{P(A_iB)}{P(B)} = \frac{P(B \mid A_i)P(A_i)}{\sum_{k=1}^{n} P(B \mid A_k)P(A_k)} \tag{3.11}$$

Bayes' rule is the basis for pattern recognition that is described in Chapter 4.

3.1.2. Random Variables

Elements in a sample space may be numbered and referred to by the numbers given. A variable X that specifies the numerical quantity in a sample space is called a *random variable*. Therefore, a random variable X is a function that maps each possible outcome s in the sample space S onto real numbers $X(s)$. Since each event is a subset of the sample space, an event is represented as a set of $\{s\}$ which satisfies $\{s \mid X(s)=x\}$. We use capital letters to denote random variables and lower-case letters to denote fixed values of the random variable. Thus, the probability that $X=x$ is denoted as:

$$P(X=x)=P(s \mid X(s)=x) \tag{3.12}$$

A random variable X is a *discrete* random variable, or X has a *discrete distribution*, if X can take only a finite number n of different values $x_1,x_2,\cdots,x_n$, or at most, an infinite sequence of different values $x_1,x_2,\cdots$. If the random variable X is a discrete random variable, the *probability function* (pf) or *probability mass function* (pmf) of X is defined to be the function p such that for any real number x,

$$p_X(x)=P(X=x) \tag{3.13}$$

For the cases in which there is no confusion, we drop the subscription X for $p_X(x)$. The sum of probability mass over all values of the random variable is equal to unity.

$$\sum_{i=1}^{n} p(x_i)=\sum_{i=1}^{n} P(X=x_i)=1 \tag{3.14}$$

The marginal probability, chain rule and Bayes' rule can also be rewritten with respect to random variables:

$$p_X(x_i)=P(X=x_i)=\sum_{k=1}^{m} P(X=x_i,Y=y_k)=\sum_{k=1}^{m} P(X=x_i \mid Y=y_k)P(Y=y_k) \tag{3.15}$$

$$\begin{aligned} &P(X_1=x_1,\cdots,X_n=x_n)= \\ &P(X_n=x_n \mid X_1=x_1,\cdots,X_{n-1}=x_{n-1})\cdots P(X_2=x_2 \mid X_1=x_1)P(X_1=x_1) \end{aligned} \tag{3.16}$$

$$P(X=x_i \mid Y=y)=\frac{P(X=x_i,Y=y)}{P(Y=y)}=\frac{P(Y=y \mid X=x_i)P(X=x_i)}{\sum_{k=1}^{n} P(Y=y \mid X=x_k)P(X=x_k)} \tag{3.17}$$

In a similar manner, if the random variables X and Y are statistically independent, they can be represented as:

$$P(X=x_i,Y=y_j)=P(X=x_i)P(Y=y_j)=p_X(x_i)p_Y(y_j) \ \forall \text{ all } i \text{ and } j \tag{3.18}$$

A random variable X is a *continuous* random variable, or X has a *continuous distribution*, if there exists a nonnegative function f, defined on the real line, such that for an interval A,

$$P(X \in A) = \int_A f_X(x)dx \tag{3.19}$$

The function f_X is called the *probability density function* (abbreviated pdf) of X. We drop the subscript X for f_X if there is no ambiguity. As illustrated in Figure 3.3, the area of shaded region is equal to the value of $P(a \le X \le b)$.

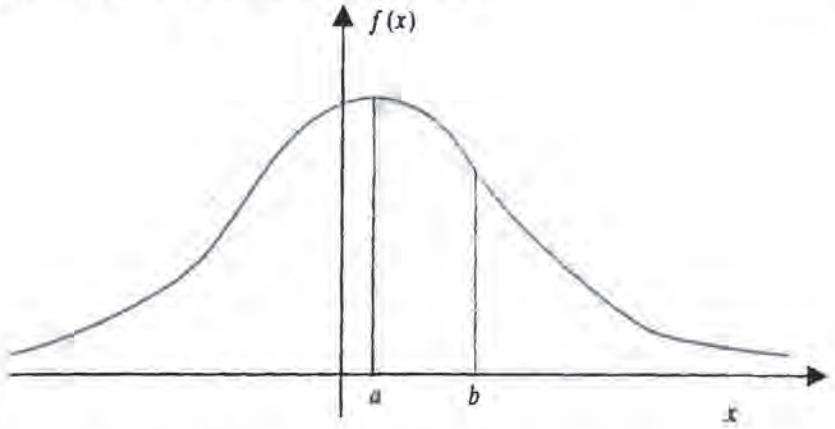


Figure 3.3 An example of pdf. The area of the shaded region is equal to the value of $P(a \le X \le b)$.

Every pdf must satisfy the following two requirements:

$$\begin{aligned} &f(x) \ge 0 \text{ for } -\infty \le x \le \infty \text{ and} \\ &\int_{-\infty}^{\infty} f(x)dx = 1 \end{aligned} \tag{3.20}$$

The marginal probability, chain rule, and Bayes' rule can also be rewritten with respect to continuous random variables:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dy = \int_{-\infty}^{\infty} f_{X|Y}(x \mid y) f_Y(y)dy \tag{3.21}$$

$$f_{X_1,\cdots,X_n}(x_1,\cdots,x_n) = f_{X_n|X_1,\cdots,X_{n-1}}(x_n \mid x_1,\cdots,x_{n-1}) \cdots f_{X_2|X_1}(x_2 \mid x_1) f_{X_1}(x_1) \tag{3.22}$$

$$f_{X|Y}(x \mid y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} = \frac{f_{Y|X}(y \mid x) f_X(x)}{\int_{-\infty}^{\infty} f_{Y|X}(y \mid x) f_X(x)dx} \tag{3.23}$$

The *distribution function* or *cumulative distribution function* F of a discrete or continuous random variable X is a function defined for every real number x as follows:

$$F(x) = P(X \leq x) \quad \text{for } -\infty \leq x \leq \infty \tag{3.24}$$

For continuous random variables, it follows that:

$$F(x) = \int_{-\infty}^{x} f_X(x)dx \tag{3.25}$$

$$f_X(x) = \frac{dF(x)}{dx} \tag{3.26}$$

3.1.3. Mean and Variance

Suppose that a discrete random variable X has a pf $f(x)$; the *expectation* or *mean* of X is defined as follows:

$$E(X) = \sum_{x} xf(x) \tag{3.27}$$

Similarly, if a continuous random variable X has a pdf f, the *expectation* or *mean* of X is defined as follows:

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx \tag{3.28}$$

In physics, the mean is regarded as the center of mass of the probability distribution. The expectation can also be defined for any function of the random variable X. If X is a continuous random variable with pdf f, then the expectation of any function $g(X)$ can be defined as follows:

$$E\left[g(X)\right] = \int_{-\infty}^{\infty} g(x)f(x)dx \tag{3.29}$$

The expectation of a random variable is a linear operator. That is, it satisfies both additivity and homogeneity properties:

$$E(a_1X_1 + \cdots + a_nX_n + b) = a_1E(X_1) + \cdots + a_nE(X_n) + b \tag{3.30}$$

where $a_1, \cdots, a_n, b$ are constants.

Equation (3.30) is valid regardless of whether or not the random variables $X_1, \cdots, X_n$ are independent.

Suppose that X is a random variable with mean $\mu = E(X)$. The *variance* of X denoted as $Var(X)$ is defined as follows:

$$Var(X) = \sigma^2 = E\left[(X-\mu)^2\right] \tag{3.31}$$

where σ, the nonnegative square root of the variance is known as the *standard deviation* of random variable X. Therefore, the variance is also often denoted as σ^2.

The variance of a distribution provides a measure of the spread or dispersion of the distribution around its mean μ. A small value of the variance indicates that the probability distribution is tightly concentrated around μ, and a large value of the variance typically indicates the probability distribution has a wide spread around μ. Figure 3.4 illustrates three different Gaussian distributions[1] with the same mean, but different variances.

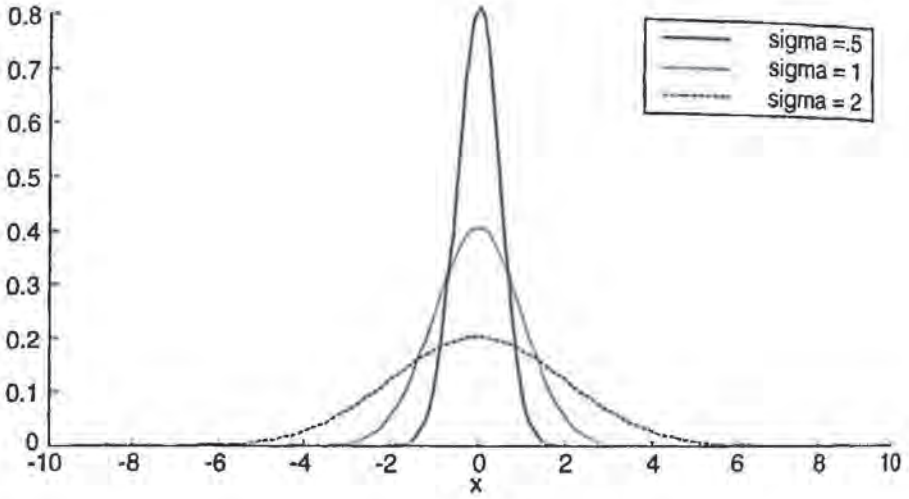


Figure 3.4 Three Gaussian distributions with same mean μ, but different variances, 0.5, 1.0, and 2.0, respectively. The distribution with a large value of the variance has a wide spread around the mean μ.

The variance of random variable X can be computed in the following way:

$$Var(X) = E(X^2) - \left[E(X)\right]^2 \tag{3.32}$$

In physics, the expectation $E(X^k)$ is called the k^{th} moment of X for any random variable X and any positive integer k. Therefore, the variance is simply the difference between the second moment and the square of the first moment.

The variance satisfies the following additivity property, if random variables X and Y are independent:

$$Var(X+Y) = Var(X) + Var(Y) \tag{3.33}$$

However, it does not satisfy the homogeneity property. Instead for constant a,

$$Var(aX) = a^2 Var(X) \tag{3.34}$$

[1] We describe Gaussian distributions in Section 3.1.7.

Since it is clear that $Var(b) = 0$ for any constant b, we have an equation similar to Eq. (3.30) if random variables $X_1, \cdots, X_n$ are independent.

$$Var(a_1X_1 + \cdots + a_nX_n + b) = a_1^2Var(X_1) + \cdots + a_n^2Var(X_n) \tag{3.35}$$

Conditional expectation can also be defined in a similar way. Suppose that X and Y are discrete random variables and let $f(y \mid x)$ denote the conditional pf of Y given $X = x$, then the conditional expectation $E(Y \mid X)$ is defined to be the function of X whose value $E(Y \mid x)$ when $X = x$ is

$$E_{Y|X}(Y \mid X = x) = \sum_y y f_{Y|X}(y \mid x) \tag{3.36}$$

For continuous random variables X and Y with $f_{Y|X}(y \mid x)$ as the conditional pdf. of Y given $X = x$, the conditional expectation $E(Y \mid X)$ is defined to be the function of X whose value $E(Y \mid x)$ when $X = x$ is

$$E_{Y|X}(Y \mid X = x) = \int_{-\infty}^{\infty} y f_{Y|X}(y \mid x) dy \tag{3.37}$$

Since $E(Y \mid X)$ is a function of random variable X, it itself is a random variable whose probability distribution can be derived from the distribution of X. It can be shown that

$$E_X\left[E_{Y|X}(Y \mid X)\right] = E_{X,Y}(Y) \tag{3.38}$$

More generally, suppose that X and Y have a continuous joint distribution and that $g(x, y)$ is any arbitrary function of X and Y. The conditional expectation $E[g(X,Y) \mid X]$ is defined to be the function of X whose value $E[g(X,Y) \mid x]$ when $X = x$ is

$$E_{Y|X}\left[g(X,Y) \mid X = x\right] = \int_{-\infty}^{\infty} g(x,y) f_{Y|X}(y \mid x) dy \tag{3.39}$$

Equation (3.38) can also be generalized into the following equation:

$$E_X\left\{E_{Y|X}\left[g(X,Y) \mid X\right]\right\} = E_{X,Y}\left[g(X,Y)\right] \tag{3.40}$$

Finally, it is worthwhile to introduce *median* and *mode*. The median of a distribution of X is defined to be a point m, such that $P(X \le m) \ge 1/2$ and $P(X \ge m) \ge 1/2$. Thus, the median m divides the total probability into two equal parts, i.e., the probability to the left of m and the probability to the right of m are exactly $1/2$.

Suppose a random variable X has either a discrete distribution with pf $p(x)$ or continuous pdf $f(x)$; a point ϖ is called the mode of the distribution if $p(x)$ or $f(x)$ attains the maximum value at the point ϖ. A distribution can have more than one mode.

3.1.3.1. The Law of Large Numbers

The concept of *sample mean* and *sample variance* is important in statistics because most statistical experiments involve sampling. Suppose that the random variables $X_1,\cdots,X_n$ form a random sample of size n from some distribution for which the mean is μ and the variance is σ^2. In other words, the random variables $X_1,\cdots,X_n$ are *independent identically distributed* (often abbreviated by iid) and each has mean μ and variance σ^2. Now if we denote $\bar{X}_n$ as the arithmetic average of the n observations in the sample, then

$$\bar{X}_n = \frac{1}{n}(X_1 + \cdots + X_n) \tag{3.41}$$

$\bar{X}_n$ is a random variable and is referred to as *sample mean*. The mean and variance of $\bar{X}_n$ can be easily derived based on the definition.

$$E(\bar{X}_n) = \mu \quad \text{and} \quad \text{Var}(\bar{X}_n) = \frac{\sigma^2}{n} \tag{3.42}$$

Equation (3.42) states that the mean of sample mean is equal to mean of the distribution, while the variance of sample mean is only $1/n$ times the variance of the distribution. In other words, the distribution of $\bar{X}_n$ will be more concentrated around the mean μ than was the original distribution. Thus, the sample mean is closer to μ than is the value of just a single observation X_i from the given distribution.

The *law of large numbers* is one of most important theorems in probability theory. Formally, it states that the sample mean $\bar{X}_n$ converges to the mean μ in probability, that is,

$$\lim_{n\to\infty} P\left(|\bar{X}_n - \mu| < \varepsilon\right) = 1 \quad \text{for any given number } \varepsilon > 0 \tag{3.43}$$

The law of large numbers basically implies that the sample mean is an excellent estimate of the unknown mean of the distribution when the sample size n is large.

3.1.4. Covariance and Correlation

Let X and Y be random variables having a specific joint distribution, and $E(X) = \mu_X$, $E(Y) = \mu_Y$, $Var(X) = \sigma_X^2$, and $Var(Y) = \sigma_Y^2$. The *covariance* of X and Y, denoted as $Cov(X,Y)$, is defined as follows:

$$Cov(X,Y) = E\left[(X-\mu_X)(Y-\mu_Y)\right] = Cov(Y,X) \tag{3.44}$$

In addition, the *correlation coefficient* of X and Y, denoted as ρ_{XY}, is defined as follows:

$$\rho_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{3.45}$$

It can be shown that $\rho(X,Y)$ should be bound within $[-1...1]$, that is,

$$-1 \le \rho(X,Y) \le 1 \tag{3.46}$$

X and Y are said to be *positively correlated* if $\rho_{XY} > 0$, *negatively correlated* if $\rho_{XY} < 0$, and *uncorrelated* if $\rho_{XY} = 0$. It can also be shown that $Cov(X,Y)$ and ρ_{XY} must have the same sign; that is, both are positive, negative, or zero at the same time. When $E(XY) = 0$, the two random variables are called *orthogonal.*

There are several theorems pertaining to the basic properties of covariance and correlation. We list here the most important ones:

Theorem 1 For any random variables X and Y

$$Cov(X,Y) = E(XY) - E(X)E(Y) \tag{3.47}$$

Theorem 2 If X and Y are independent random variables, then

$$Cov(X,Y) = \rho_{XY} = 0$$

Theorem 3 Suppose X is a random variable and Y is a linear function of X in the form of $Y = aX + b$ for some constant a and b, where $a \neq 0$. If $a > 0$, then $\rho_{XY} = 1$. If $a < 0$, then $\rho_{XY} = -1$. Sometimes, ρ_{XY} is referred to as the amount of linear dependency between random variables X and Y.

Theorem 4 For any random variables X and Y,

$$Var(X+Y) = Var(X) + Var(Y) + 2Cov(X,Y) \tag{3.48}$$

Theorem 5 If $X_1, \cdots, X_n$ are random variables, then

$$Var(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} Var(X_i) + 2\sum_{i=1}^{n}\sum_{j=1}^{i-1} Cov(X_i, X_j) \tag{3.49}$$

3.1.5. Random Vectors and Multivariate Distributions

When a random variable is a vector rather than a scalar, it is called a random vector and we often use boldface variable like $\mathbf{X} = (X_1, \cdots, X_n)$ to indicate that it is a random vector. It is said that n random variables $X_1, \cdots, X_n$ have a *discrete joint distribution* if the random vector $\mathbf{X} = (X_1, \cdots, X_n)$ can have only a finite number or an infinite sequence of different

values $(x_1,\cdots,x_n)$ in R^n. The joint pf of $X_1,\cdots,X_n$ is defined to be the function f_X such that for any point $(x_1,\cdots,x_n)\in R^n$,

$$f_X(x_1,\cdots,x_n)=P(X_1=x_1,\cdots,X_n=x_n) \tag{3.50}$$

Similarly, it is said that n random variables $X_1,\cdots,X_n$ have a *continuous joint distribution* if there is a nonnegative function f defined on R^n such that for any subset $A\subset R^n$,

$$P[(X_1,\cdots,X_n)\in A]=\int_A\cdots\int f_X(x_1,\cdots,x_n)dx_1\cdots dx_n \tag{3.51}$$

The *joint distribution function* can also be defined similarly for n random variables $X_1,\cdots,X_n$ as follows:

$$F_X(x_1,\cdots,x_n)=P(X_1\le x_1,\cdots,X_n\le x_n) \tag{3.52}$$

The concept of mean and variance for a random vector can be generalized into *mean vector* and *covariance matrix*. Supposed that $\mathbf{X}$ is an n-dimensional random vector with components $X_1,\cdots,X_n$, under matrix representation, we have

$$\mathbf{X}=\begin{bmatrix}X_1\\ \vdots\\ X_n\end{bmatrix} \tag{3.53}$$

The *expectation (mean) vector* $E(\mathbf{X})$ of random vector $\mathbf{X}$ is an n-dimensional vector whose components are the expectations of the individual components of $\mathbf{X}$, that is,

$$E(\mathbf{X})=\begin{bmatrix}E(X_1)\\ \vdots\\ E(X_n)\end{bmatrix} \tag{3.54}$$

The *covariance matrix* $Cov(\mathbf{X})$ of random vector $\mathbf{X}$ is defined to be an $n\times n$ matrix such that the element in the i^{th} row and j^{th} column is $Cov(X_i,Y_j)$, that is,

$$Cov(\mathbf{X})=\begin{bmatrix}Cov(X_1,X_1) & \cdots & Cov(X_1,X_n)\\ \vdots & & \vdots\\ Cov(X_n,X_1) & \cdots & Cov(X_n,X_n)\end{bmatrix}=E\left[[X-E(X)][X-E(X)]^t\right] \tag{3.55}$$

It should be emphasized that the n diagonal elements of the covariance matrix $Cov(\mathbf{X})$ are actually the variances of $X_1,\cdots,X_n$. Furthermore, since the covariance is symmetric, i.e., $Cov(X_i,X_j)=Cov(X_j,X_i)$, the covariance matrix $Cov(\mathbf{X})$ must be a symmetric matrix.

There is an important theorem regarding the mean vector and covariance matrix for a linear transformation of random vector $\mathbf{X}$. Suppose $\mathbf{X}$ is an n-dimensional vector as specified by Eq. (3.53), with mean vector $E(\mathbf{X})$ and covariance matrix $Cov(\mathbf{X})$. Now, assume $\mathbf{Y}$ is a m-dimensional random vector which is a linear transform of random vector $\mathbf{X}$ by the

relation: $\mathbf{Y}=\mathbf{AX}+\mathbf{B}$, where $\mathbf{A}$ is a $m\times n$ transformation matrix whose elements are constants, and $\mathbf{B}$ is a m-dimensional constant vector. Then we have the following two equations:

$$E(\mathbf{Y})=\mathbf{A}E(\mathbf{X})+\mathbf{B} \tag{3.56}$$

$$Cov(\mathbf{Y})=\mathbf{A}Cov(\mathbf{X})\mathbf{A}' \tag{3.57}$$

3.1.6. Some Useful Distributions

In the following two sections, we will introduce several useful distributions that are widely used in applications of probability and statistics, particularly in spoken language systems.

3.1.6.1. Uniform Distributions

The simplest distribution is uniform distribution where the pf or pdf is a constant function. For uniform discrete random variable X, which only takes possible values from $\{x_i \mid 1\le i\le n\}$, the pf for X is

$$P(X=x_i)=\frac{1}{n} \quad 1\le i\le n \tag{3.58}$$

For uniform continuous random variable X, which only takes possible values in the real interval $[a,b]$, as shown in Figure 3.5, the pdf for X is

$$f(x)=\frac{1}{b-a} \quad a\le x\le b \tag{3.59}$$

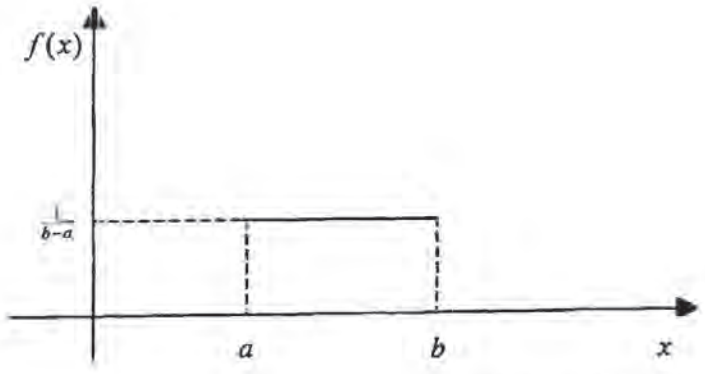


Figure 3.5 A uniform distribution for pdf in Eq. (3.59).

3.1.6.2. Binomial Distributions

The *binomial distribution* is used to describe binary-decision events. For example, suppose that a single coin toss will produce heads with probability p and produce tails with probability $1-p$. Now, if we toss the same coin n times and let X denote the number of heads observed, then the random variable X has the following *binomial* pf:

$$P(X=x)=f(x\mid n,p)=\binom{n}{x}p^x(1-p)^{n-x} \tag{3.60}$$

It can be shown that the mean and variance of a binomial distribution are:

$$E(X)=np \tag{3.61}$$

$$Var(X)=np(1-p) \tag{3.62}$$

Figure 3.6 illustrates three binomial distributions with p = 0.2, 0.3, and 0.4, and n = 10.

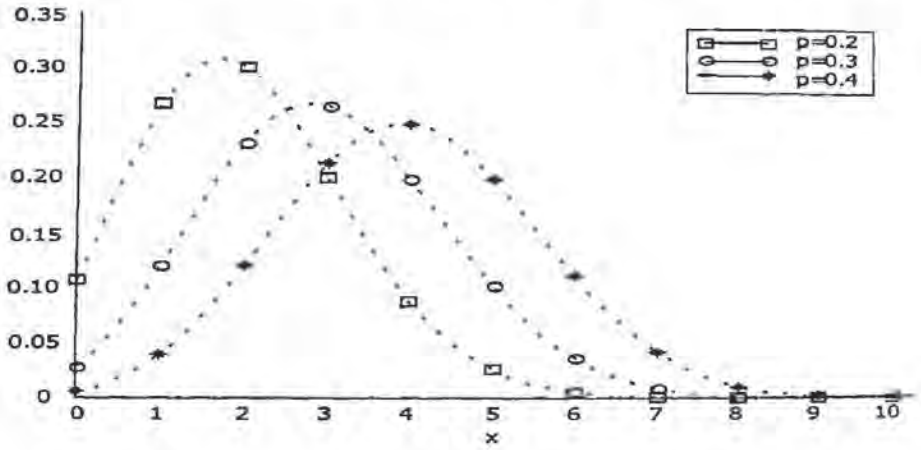


Figure 3.6 Three binomial distributions with p = 0.2, 0.3, and 0.4, and n = 10.

3.1.6.3. Geometric Distributions

The geometric distribution is related to the binomial distribution. As in the independent coin toss example, heads has a probability p and tails has a probability $1-p$. The geometric distribution is to model the time until tails appears. Let the random variable X

be the time (the number of tosses) until the first tail-up is shown. The pdf of X is in the following form:

$$P(X=x)=f(x\mid p)=p^{x-1}(1-p) \quad x=1,2,\ldots \text{ and } 0<p<1 \tag{3.63}$$

The mean and variance of a geometric distribution are given by:

$$E(X)=\frac{1}{1-p} \tag{3.64}$$

$$Var(X)=\frac{1}{(1-p)^2} \tag{3.65}$$

One example for the geometric distribution is the distribution of the state duration for a hidden Markov model, as described in Chapter 8. Figure 3.7 illustrates three geometric distributions with $p = 0.1$, 0.4, and 0.7.

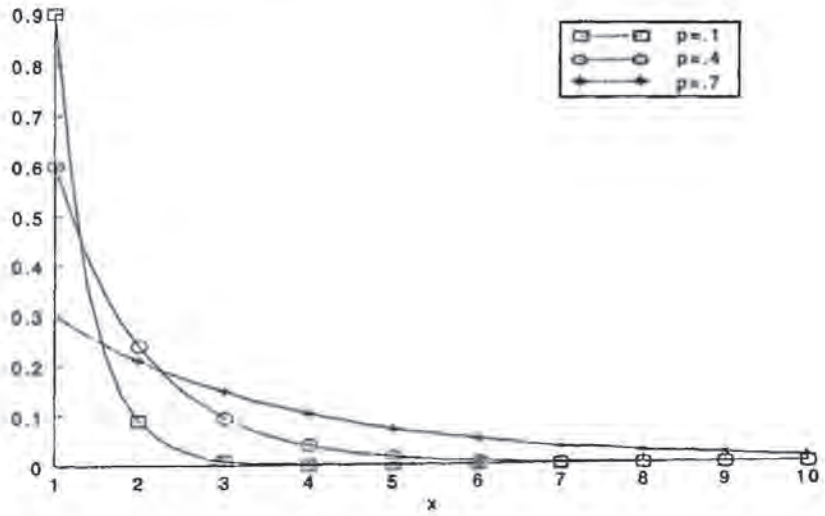


Figure 3.7 Three geometric distributions with different parameter p.

3.1.6.4. Multinomial Distributions

Suppose that a bag contains balls of k different colors, where the proportion of the balls of color i is p_i. Thus, $p_i > 0$ for $i=1,\ldots,k$ and $\sum_{i=1}^{k} p_i = 1$. Now suppose that n balls are randomly selected from the bag and there are enough balls ($> n$) of each color. Let X_i denote

the number of selected balls that are of color i. The random vector $\mathbf{X}=(X_1,\ldots,X_k)$ is said to have a *multinomial distribution* with parameters n and $\mathbf{p}=(p_1,\ldots,p_k)$. For a vector $\mathbf{x}=(x_1,\ldots,x_k)$, the pf of $\mathbf{X}$ has the following form:

$$P(\mathbf{X}=\mathbf{x})=f(\mathbf{x}\mid n,\mathbf{p})=\begin{cases}\dfrac{n!}{x_1!\ldots x_k!}p_1^{x_1}\ldots p_k^{x_k} & \text{where } x_i\geq 0\ \forall i=1\ \ldots\ k \\ & \text{and } x_1+\cdots+x_k=n \\ 0 & \text{otherwise}\end{cases} \tag{3.66}$$

It can be shown that the mean, variance and covariance of the multinomial distribution are:

$$E(X_i)=np_i \quad \text{and} \quad Var(X_i)=np_i(1-p_i) \quad \forall i=1,\ldots,k \tag{3.67}$$

$$Cov(X_i,X_j)=-np_ip_j \tag{3.68}$$

Figure 3.8 shows a multinomial distribution with $n=10$, $p_1=0.2$, and $p_2=0.3$. Since there are only two free parameters x_1 and x_2, the graph is illustrated only using x_1

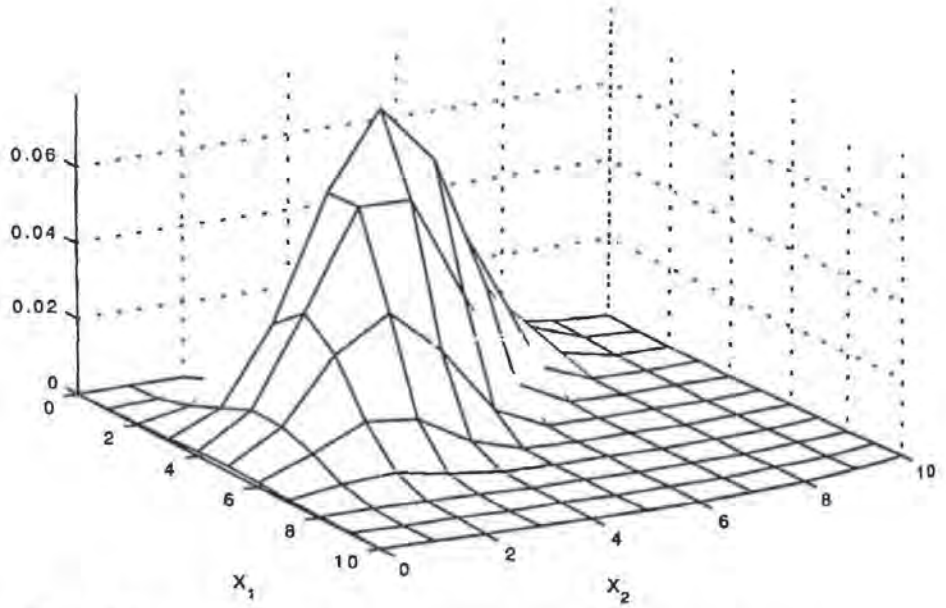


Figure 3.8 A multinomial distribution with n=10, $p_1=0.2$, and $p_2=0.3$.

and x_2 as axis. Multinomial distributions are typically used with the χ^2 test that is one of the most widely used goodness-of-fit hypotheses testing procedures described in Section 3.3.3.

3.1.6.5. Poisson Distributions

Another popular discrete distribution is the *Poisson distribution*. The random variable X has a Poisson distribution with mean λ ($\lambda > 0$) if the pf of X has the following form:

$$P(X = x) = f(x \mid \lambda) = \begin{cases} \dfrac{e^{-\lambda}\lambda^x}{x!} & \text{for } x = 0, 1, 2, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{3.69}$$

The mean and variance of a Poisson distribution are the same and equal λ:

$$E(X) = Var(X) = \lambda \tag{3.70}$$

Figure 3.9 illustrates three Poisson distributions with $\lambda = 1$, 2, and 4. The Poisson distribution is typically used in queuing theory, where x is the total number of occurrences of some phenomenon during a fixed period of time or within a fixed region of space. Examples include the number of telephone calls received at a switchboard during a fixed period of time. In speech recognition, the Poisson distribution is used to model the duration for a phoneme.

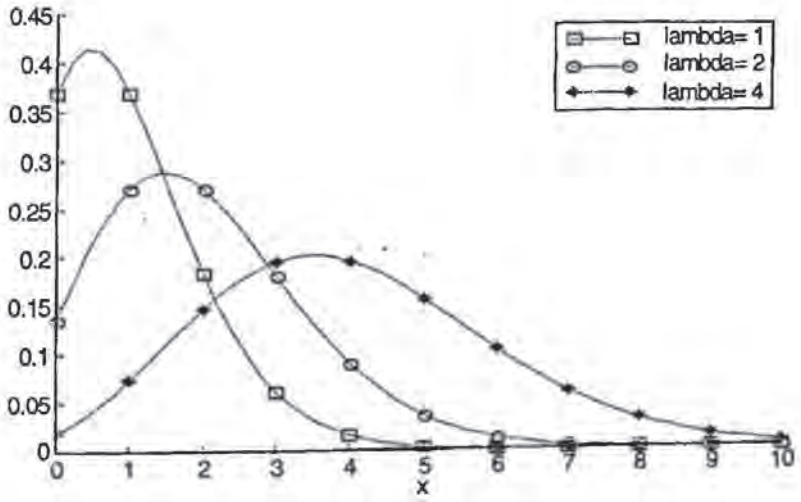


Figure 3.9 Three Poisson distributions with $\lambda = 1$, 2, and 4.

3.1.6.6. Gamma Distributions

A continuous random variable X is said to have a *gamma distribution* with parameters α and β ($\alpha > 0$ and $\beta > 0$) if X has a continuous pdf of the following form:

$$f(x \mid \alpha, \beta) = \begin{cases} \dfrac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} & x > 0 \\ 0 & x \le 0 \end{cases} \tag{3.71}$$

where

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x} dx \tag{3.72}$$

It can be shown that the function Γ is a factorial function when α is a positive integer.

$$\Gamma(n) = \begin{cases} (n-1)! & n = 2,3,... \\ 1 & n = 1 \end{cases} \tag{3.73}$$

The mean and variance of a gamma distribution are:

$$E(X) = \frac{\alpha}{\beta} \text{ and } Var(X) = \frac{\alpha}{\beta^2} \tag{3.74}$$

Figure 3.10 illustrates three gamma distributions with $\beta = 1.0$ and $\alpha = 2.0$, 3.0, and 4.0. There is an interesting theorem associated with gamma distributions. If the random variables $X_1, \ldots, X_k$ are independent and each random variable X_i has a gamma distribution with parameters α_i and β, then the sum $X_1 + \cdots + X_k$ also has a gamma distribution with parameters $\alpha_1 + \cdots + \alpha_k$ and β.

A special case of gamma distribution is called *exponential distribution*. A continuous random variable X is said to have an *exponential distribution* with parameters β ($\beta > 0$) if X has a continuous pdf of the following form:

$$f(x \mid \beta) = \begin{cases} \beta e^{-\beta x} & x > 0 \\ 0 & x \le 0 \end{cases} \tag{3.75}$$

It is clear that the exponential distribution is a gamma distribution with $\alpha = 1$. The mean and variance of the exponential distribution are:

$$E(X) = \frac{1}{\beta} \text{ and } Var(X) = \frac{1}{\beta^2} \tag{3.76}$$

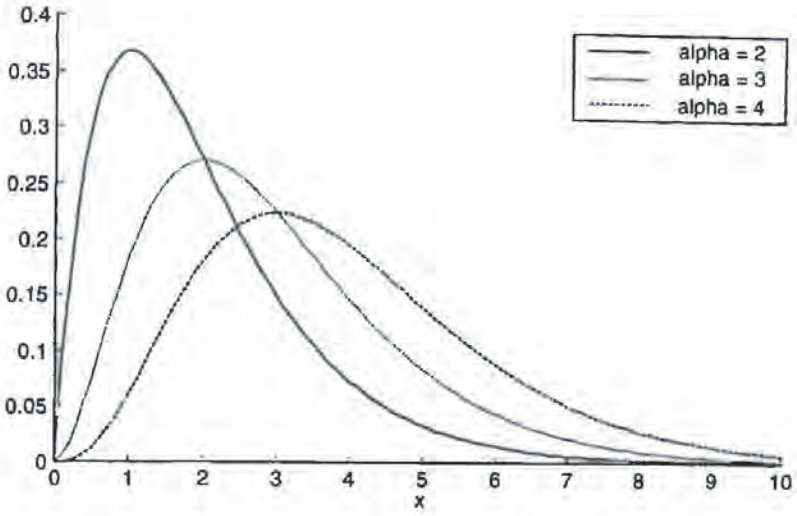


Figure 3.10 Three Gamma distributions with $\beta = 1.0$ and $\alpha = 2.0$, 3.0, and 4.0.

Figure 3.11 shows three exponential distributions with $\beta = 1.0$, 0.6, and 0.3. The exponential distribution is often used in queuing theory for the distributions of the duration of a service or the inter-arrival time of customers. It is also used to approximate the distribution of the life of a mechanical component.

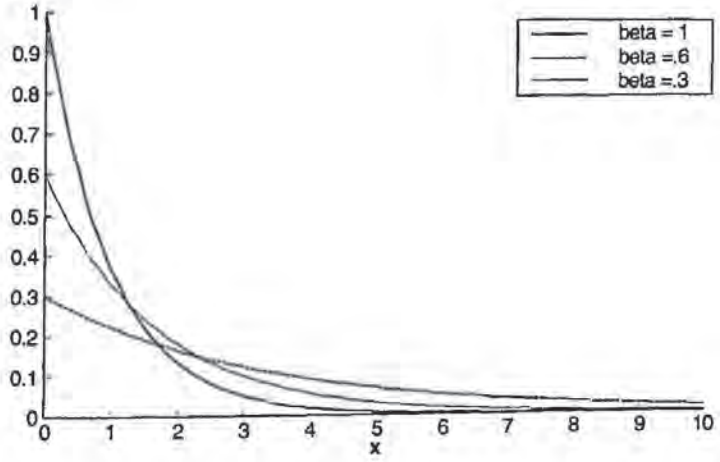


Figure 3.11 Three exponential distributions with $\beta = 1.0$, 0.6, and 0.3.

3.1.7. Gaussian Distributions

Gaussian distribution is by far the most important probability distribution mainly because many scientists have observed that the random variables studied in various physical experiments (including speech signals), often have distributions that are approximately Gaussian. The Gaussian distribution is also referred to as normal distribution. A continuous random variable X is said to have a *Gaussian distribution* with mean μ and variance σ^2 $(\sigma > 0)$ if X has a continuous pdf in the following form:

$$f(x \mid \mu,\sigma^2) = N(\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{3.77}$$

It can be shown that μ and σ^2 are indeed the mean and the variance for the Gaussian distribution. Some examples of Gaussians can be found in Figure 3.4.

The use of Gaussian distributions is justified by the *Central Limit Theorem*, which states that observable events considered to be a consequence of many unrelated causes with no single cause predominating over the others, tend to follow the Gaussian distribution [6].

It can be shown from Eq. (3.77) that the Gaussian $f(x \mid \mu,\sigma^2)$ is symmetric with respect to $x = \mu$. Therefore, μ is both the mean and the median of the distribution. Moreover, μ is also the mode of the distribution, i.e., the pdf $f(x \mid \mu,\sigma^2)$ attains its maximum at the mean point $x = \mu$.

Several Gaussian pdfs with the same mean μ, but different variances are illustrated in Figure 3.4. Readers can see that the curve has a *bell* shape. The Gaussian pdf with a small variance has a high peak and is very concentrated around the mean μ, whereas the Gaussian pdf with a large variance is relatively flat and is spread out more widely over the x-axis.

If the random variable X is a *Gaussian distribution* with mean μ and variance σ^2, then any linear function of X also has a Gaussian distribution. That is, if $Y = aX + b$, where a and b are constants and $a \neq 0$, Y has a Gaussian distribution with mean $a\mu + b$ and variance $a^2\sigma^2$. Similarly, the sum $X_1 + \cdots + X_k$ of random variables $X_1,\ldots,X_k$, where each random variable X_i has a Gaussian distribution, is also a Gaussian distribution.

3.1.7.1. Standard Gaussian Distributions

The Gaussian distribution with mean 0 and variance 1, denoted as $N(0,1)$, is called the *standard Gaussian distribution* or *unit Gaussian distribution*. Since the linear transformation of a Gaussian distribution is still a Gaussian distribution, the behavior of a Gaussian distribution can be solely described using a standard Gaussian distribution. If the random variable X is a Gaussian distribution with mean μ and variance σ^2, that is, $X \sim N(\mu,\sigma^2)$, it can be shown that

$$Z = \frac{X-\mu}{\sigma} \sim N(0,1) \tag{3.78}$$

Based on Eq. (3.78), the following property can be shown:

$$P(|X-\mu| \le k\sigma) = P(|Z| \le k) \tag{3.79}$$

Equation (3.79) demonstrates that every Gaussian distribution contains the same total amount of probability within any fixed number of standard deviations of its mean.

3.1.7.2. The Central Limit Theorem

If random variables $X_1,\ldots,X_n$ are i.i.d. according to a common distribution function with mean μ and variance σ^2, then as the random sample size n approaches ∞, the following random variable has a distribution converging to the standard Gaussian distribution:

$$Y_n = \frac{n(\bar{X}_n - \mu)}{\sqrt{n\sigma^2}} \sim N(0,1) \tag{3.80}$$

where $\bar{X}_n$ is the sample mean of random variables $X_1,\ldots,X_n$ as defined in Eq. (3.41).

Based on Eq. (3.80), the sample mean random variable $\bar{X}_n$ can be approximated by a Gaussian distribution with mean μ and variance σ^2/n.

The central limit theorem above is applied to i.i.d. random variables $X_1,\ldots,X_n$. A. Liapounov in 1901 derived another central limit theorem for independent but not necessarily identically distributed random variables $X_1,\ldots,X_n$. Suppose $X_1,\ldots,X_n$ are independent random variables and $E(|X_i-\mu_i|^3) < \infty$ for $1 \le i \le n$; the following random variable will converge to standard Gaussian distribution when $n \to \infty$.

$$Y_n = \left(\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \mu_i\right) \Big/ \left(\sum_{i=1}^{n} \sigma_i^2\right)^{1/2} \tag{3.81}$$

In other words, the sum of random variables $X_1,\ldots,X_n$ can be approximated by a Gaussian distribution with mean $\sum_{i=1}^{n} \mu_i$ and variance $\left(\sum_{i=1}^{n} \sigma_i^2\right)^{1/2}$.

Both central limit theorems essentially state that regardless of their original individual distributions, the sum of many independent random variables (effects) tends to be distributed like a Gaussian distribution as the number of random variables (effects) becomes large.

3.1.7.3. Multivariate Mixture Gaussian Distributions

When $\mathbf{X} = (X_1,\ldots,X_n)$ is an n-dimensional continuous random vector, the multivariate Gaussian pdf has the following form:

$$f(\mathbf{X}=\mathbf{x}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}) = N(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{3.82}$$

where $\boldsymbol{\mu}$ is the n-dimensional mean vector, $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix, and $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix $\boldsymbol{\Sigma}$.

$$\boldsymbol{\mu} = E(\mathbf{x}) \tag{3.83}$$

$$\boldsymbol{\Sigma} = E\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})'\right] \tag{3.84}$$

More specifically, the i-j^{th} element σ_{ij}^2 of covariance matrix $\boldsymbol{\Sigma}$ can be specified as follows:

$$\sigma_{ij}^2 = E\left[(x_i - \mu_i)(x_j - \mu_j)\right] \tag{3.85}$$

If $X_1, \ldots, X_n$ are independent random variables, the covariance matrix $\boldsymbol{\Sigma}$ is reduced to diagonal covariance where all the off-diagonal entries are zero. The distribution can be regarded as n independent scalar Gaussian distributions. The joint pdf is the product of all the individual scalar Gaussian pdfs. Figure 3.12 shows a two-dimensional multivariate Gaussian distribution with independent random variables x_1 and x_2 with the same variance. Figure 3.13 shows another two-dimensional multivariate Gaussian distribution with independent random variables x_1 and x_2 that have different variances.

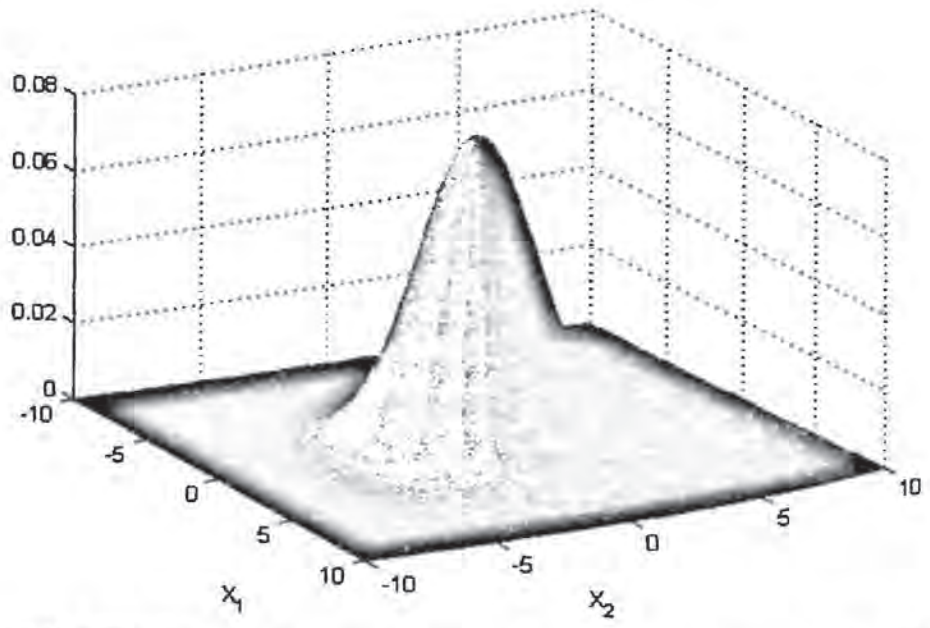


Figure 3.12 A two-dimensional multivariate Gaussian distribution with independent random variables x_1 and x_2 that have the same variance.

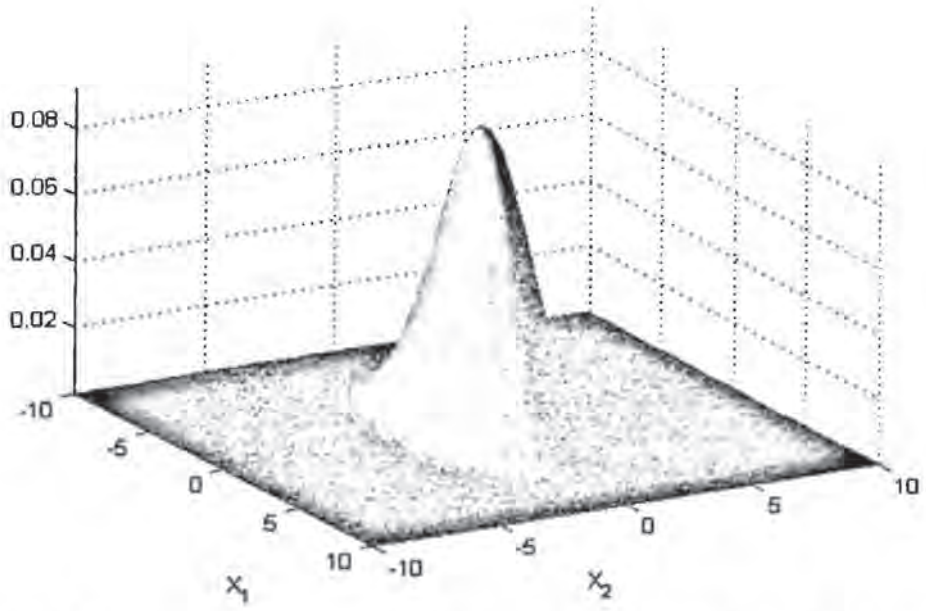


Figure 3.13 Another two-dimensional multivariate Gaussian distribution with independent random variable x_1 and x_2 which have different variances.

Although Gaussian distributions are unimodal,[2] more complex distributions with multiple local maxima can be approximated by Gaussian mixtures:

$$f(\mathbf{x}) = \sum_{k=1}^{K} c_k N_k(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.86}$$

where c_k, the mixture weight associated with kth Gaussian component, is subject to the following constraint:

$$c_k \geq 0 \text{ and } \sum_{k=1}^{K} c_k = 1$$

Gaussian mixtures with enough mixture components can approximate any distribution. Throughout this book, most continuous probability density functions are modeled with Gaussian mixtures.

3.1.7.4. χ^2 Distributions

The gamma distribution with parameters α and β is defined 1). For any given positive integer n, the gamma distribution for which $\alpha = n/2$ a is called the χ^2

[2] A unimodal distribution has a single maximum (bump) for the distribution. F ibution, the maximum occurs at the mean.

distribution with n degrees of freedom. It follows from Eq. (3.71) that the pdf for the χ^2 distribution is

$$f(x|n)=\begin{cases}\dfrac{1}{2^{n/2}\Gamma(n/2)}x^{(n/2)-1}e^{-x/2} & x>0\\ 0 & x\le 0\end{cases} \tag{3.87}$$

χ^2 distributions are important in statistics because they are closely related to random samples of Gaussian distribution. They are widely applied in many important problems of statistical inference and hypothesis testing. Specifically, if the random variables $X_1,\ldots,X_n$ are independent and identically distributed, and if each of these variables has a standard Gaussian distribution, then the sum of square $X_1^2+\ldots+X_n^2$ can be proved to have a χ^2 distribution with n degrees of freedom. Figure 3.14 illustrates three χ^2 distributions with $n=2$, 3, and 4.

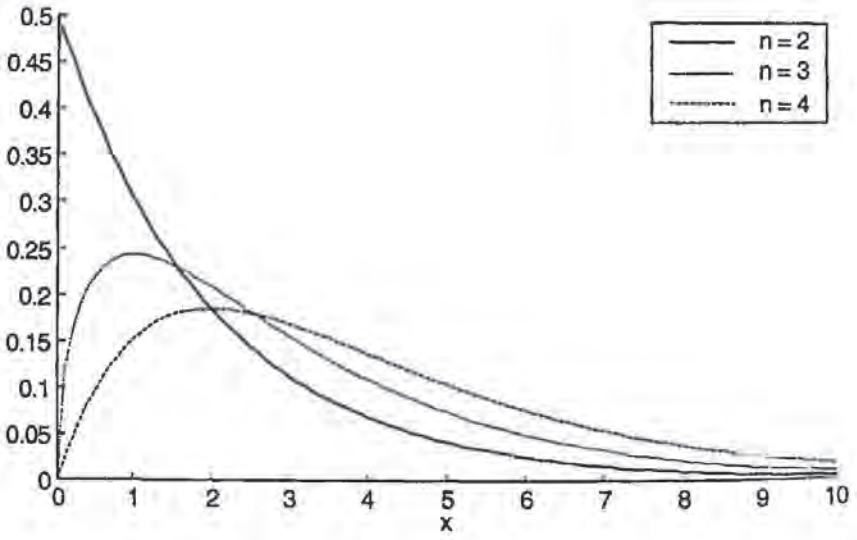


Figure 3.14 Three χ^2 distributions with $n=2$, 3, and 4.

The mean and variance for the χ^2 distribution are

$$E(X)=n \text{ and } Var(X)=2n \tag{3.88}$$

Following the additivity property of the gamma distribution, the χ^2 distribution also has the additivity property. That is, if the random variables $X_1,\ldots,X_n$ are independent and if X_i has a χ^2 distribution with k_i degrees of freedom, the sum $X_1+\ldots+X_n$ has a χ^2 distribution with $k_1+\ldots+k_n$ degrees of freedom.

3.1.7.5. Log-Normal Distribution

Let x be a Gaussian random variable with mean μ_x and standard deviation σ_x, then

$$y = e^x \tag{3.89}$$

follows the *lognormal* distribution

$$f(y \mid \mu_x, \sigma_x) = \frac{1}{y\sigma_x\sqrt{2\pi}} \exp\left\{-\frac{(\ln y - \mu_x)^2}{2\sigma_x^2}\right\} \tag{3.90}$$

shown in Figure 3.15, and whose mean is given by

$$\begin{aligned} \mu_y = E\{y\} = E\{e^x\} &= \int_{-\infty}^{\infty} \exp\{x\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-\mu_x)^2}{2\sigma_x^2}\right\} dx \\ &= \int_{-\infty}^{\infty} \exp\{\mu_x + \sigma_x^2/2\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-(\mu_x+\sigma_x^2))^2}{2\sigma_x^2}\right\} dx = \exp\{\mu_x + \sigma_x^2/2\} \end{aligned} \tag{3.91}$$

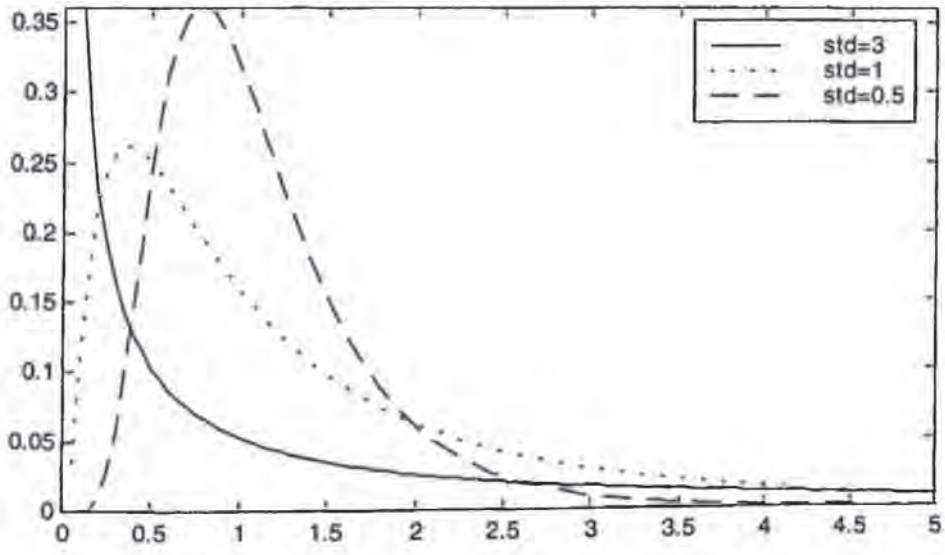


Figure 3.15 Lognormal distribution for $\mu_x = 0$ and $\sigma_x = 3$, 1, and 0.5, according to Eq. (3.90).

where we have rearranged the quadratic form of x and made use of the fact that the total probability mass of a Gaussian is 1. Similarly, the second order moment of y is given by

$$E\{y^2\} = \int_{-\infty}^{\infty} \exp\{2x\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-\mu_x)^2}{2\sigma_x^2}\right\} dx$$
$$= \int_{-\infty}^{\infty} \exp\{2\mu_x + 2\sigma_x^2\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-(\mu_x+2\sigma_x^2))^2}{2\sigma_x^2}\right\} dx = \exp\{2\mu_x + 2\sigma_x^2\} \tag{3.92}$$

and thus the variance of y is given by

$$\sigma_y^2 = E\{y^2\} - (E\{y\})^2 = \mu_y^2 \left(\exp\{\sigma_x^2\} - 1\right) \tag{3.93}$$

Similarly, if $\mathbf{x}$ is a Gaussian random vector with mean $\boldsymbol{\mu}_x$ and covariance matrix $\boldsymbol{\Sigma}_x$, then random vector $\mathbf{y} = e^{\mathbf{x}}$ is log-normal with mean and covariance matrix [8] given by

$$\boldsymbol{\mu}_y[i] = \exp\{\boldsymbol{\mu}_x[i] + \boldsymbol{\Sigma}_x[i,i]/2\}$$
$$\boldsymbol{\Sigma}_y[i,j] = \boldsymbol{\mu}_y[i]\boldsymbol{\mu}_y[j]\left(\exp\{\boldsymbol{\Sigma}_x[i,j]\} - 1\right) \tag{3.94}$$

using a similar derivation as in Eqs. (3.91) to (3.93).

3.2. ESTIMATION THEORY

Estimation theory and *significance testing* are two of the most important theories and methods of *statistical inference*. In this section, we describe estimation theory while significance testing is covered in the next section. A problem of statistical inference is one in which data generated in accordance with some unknown probability distribution must be analyzed, and some type of inference about the unknown distribution must be made. In a problem of statistical inference, any characteristic of the distribution generating the experimental data, such as the mean μ and variance σ^2 of a Gaussian distribution, is called a parameter of the distribution. The set Ω of all possible values of a *parameter* Φ or a group of parameters $\Phi_1, \Phi_2, \ldots, \Phi_n$ is called the *parameter space*. In this section we focus on how to estimate the parameter Φ from sample data.

Before we describe various estimation methods, we introduce the concept and nature of the estimation problems. Suppose that a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is iid according to a pdf $p(x|\Phi)$ where the value of the parameter Φ is unknown. Now, suppose also that the value of Φ must be estimated from the observed values in the sample. An *estimator* of the parameter Φ, based on the random variables $X_1, X_2, \ldots, X_n$, is a real-valued function $\theta(X_1, X_2, \ldots, X_n)$ that specifies the estimated value of Φ for each possible set of values of $X_1, X_2, \ldots, X_n$. That is, if the sample values of $X_1, X_2, \ldots, X_n$ turn out to be $x_1, x_2, \ldots, x_n$, then the estimated value of Φ will be $\theta(x_1, x_2, \ldots, x_n)$.

We need to distinguish between *estimator*, *estimate*, and *estimation*. An estimator $\theta(X_1, X_2, \ldots, X_n)$ is a function of the random variables, whose probability distribution can be derived from the joint distribution of $X_1, X_2, \ldots, X_n$. On the other hand, an estimate is a specific value $\theta(x_1, x_2, \ldots, x_n)$ of the estimator that is determined by using some specific

sample values $x_1, x_2, \ldots, x_n$. Estimation is usually used to indicate the process of obtaining such an estimator for the set of random variables or an estimate for the set of specific sample values. If we use the notation $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ to represent the vector of random variables and $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ to represent the vector of sample values, an estimator can be denoted as $\theta(\mathbf{X})$ and an estimate $\theta(\mathbf{x})$. Sometimes we abbreviate an estimator $\theta(\mathbf{X})$ by just the symbol θ.

In the following four sections we describe and compare three different estimators (estimation methods). They are *minimum mean square estimator*, *maximum likelihood estimator*, and *Bayes' estimator*. The first one is often used to estimate the random variable itself, while the latter two are used to estimate the parameters of the distribution of the random variables.

3.2.1. Minimum/Least Mean Squared Error Estimation

Minimum mean squared error (MMSE) estimation and *least squared error* (LSE) estimation are important methods for random variables since the goal (minimize the squared error) is an intuitive one. In general, two random variables X and Y are i.i.d. according to some pdf $f_{X,Y}(x,y)$. Suppose that we perform a series of experiments and observe the value of X. We want to find a transformation $\hat{Y} = g(X)$ such that we can predict the value of the random variable Y. The following quantity can measure the goodness of such a transformation:

$$E(Y-\hat{Y})^2 = E(Y-g(X))^2 \tag{3.95}$$

This quantity is called *mean squared error* (MSE) because it is the mean of the squared error of the predictor $g(X)$. The criterion of minimizing the mean squared error is a good one for picking the predictor $g(X)$. Of course, we usually specify the class of function G, from which $g(X)$ may be selected. In general, there is a parameter vector $\mathbf{\Phi}$ associated with the function $g(X)$, so the function can be expressed as $g(X,\mathbf{\Phi})$. The process to find the parameter vector $\hat{\mathbf{\Phi}}_{MMSE}$ that minimizes the mean of the squared error is called *minimum mean squared error estimation* and $\hat{\mathbf{\Phi}}_{MMSE}$ is called the *minimum mean squared error estimator*. That is,

$$\hat{\mathbf{\Phi}}_{MMSE} = \arg\min_{\mathbf{\Phi}} \left[E\left[(Y - g(X,\mathbf{\Phi}))^2 \right] \right] \tag{3.96}$$

Sometimes, the joint distribution of random variables X and Y is not known. Instead, samples of (x,y) pairs may be observable. In this case, the following criterion can be used instead,

$$\hat{\mathbf{\Phi}}_{LSE} = \arg\min_{\mathbf{\Phi}} \sum_{i=1}^{n} \left[y_i - g(x_i,\mathbf{\Phi}) \right]^2 \tag{3.97}$$

The argument of the minimization in Eq. (3.97) is called *sum-of-squared-error* (SSE) and the process of finding the parameter vector $\hat{\mathbf{\Phi}}_{LSE}$, which satisfies the criterion is called *least*

squared error estimation or *minimum squared error estimation*. LSE is a powerful mechanism for curve fitting, where the function $g(x,\Phi)$ describes the observation pairs (x_i, y_i). In general, there are more points (n) than the number of free parameters in function $g(x,\Phi)$, so the fitting is over-determined. Therefore, no exact solution exists, and LSE fitting becomes necessary.

It should be emphasized that MMSE and LSE are actually very similar and share similar properties. The quantity in Eq. (3.97) is actually n times the sample mean of the squared error. Based on the law of large numbers, when the joint probability $f_{X,Y}(x,y)$ is uniform or the number of samples approaches to infinity, MMSE and LSE are equivalent.

For the class of functions, we consider the following three cases:

- Constant functions, i.e.,

$$G_c = \{g(x) = c, c \in R\} \tag{3.98}$$

- Linear functions, i.e.,

$$G_l = \{g(x) = ax + b \quad a,b \in R\} \tag{3.99}$$

- Other non-linear functions G_{nl}

3.2.1.1. MMSE/LSE for Constant Functions

When $\hat{Y} = g(x) = c$, Eq. (3.95) becomes

$$E(Y - \hat{Y})^2 = E(Y - c)^2 \tag{3.100}$$

To find the MMSE estimate for c, we take the derivatives of both sides in Eq. (3.100) with respect to c and equate it to 0. The MMSE estimate c_{MMSE} is given as

$$c_{MMSE} = E(Y) \tag{3.101}$$

and the minimum mean squared error is exactly the variance of Y, $Var(Y)$.

For the LSE estimate of c, the quantity in Eq. (3.97) becomes

$$\min \sum_{i=1}^{n} [y_i - c]^2 \tag{3.102}$$

Similarly, the LSE estimate c_{LSE} can be obtained as follows:

$$c_{LSE} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{3.103}$$

The quantity in Eq. (3.103) is the sample mean.

3.2.1.2. MMSE and LSE for Linear Functions

When $\hat{Y} = g(x) = ax + b$, Eq. (3.95) becomes

$$e(a,b) = E(Y - \hat{Y})^2 = E(Y - ax - b)^2 \tag{3.104}$$

To find the MMSE estimate of a and b, we can first set

$$\frac{\partial e}{\partial a} = 0, \text{ and } \frac{\partial e}{\partial b} = 0 \tag{3.105}$$

and solve the two linear equations. Thus, we can obtain

$$a = \frac{\text{cov}(X,Y)}{Var(X)} = \rho_{XY}\frac{\sigma_Y}{\sigma_X} \tag{3.106}$$

$$b = E(Y) - \rho_{XY}\frac{\sigma_Y}{\sigma_X}E(X) \tag{3.107}$$

For LSE estimation, we assume that the sample $\mathbf{x}$ is a d-dimensional vector for generality. Assuming we have n sample-vectors $(\mathbf{x}_i, y_i) = (x_i^1, x_i^2, \cdots, x_i^d, y_i)$, $i = 1 \ldots n$, a linear function can be represented as

$$\hat{\mathbf{Y}} = \mathbf{XA} \text{ or } \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1^1 & \cdots & x_1^d \\ 1 & x_2^1 & \cdots & x_2^d \\ \vdots & \vdots & & \vdots \\ 1 & x_n^1 & \cdots & x_n^d \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} \tag{3.108}$$

The sum of squared error can then be represented as

$$e(\mathbf{A}) = \| \hat{\mathbf{Y}} - \mathbf{Y} \|^2 = \sum_{i=1}^{n} \left(\mathbf{A}^t \mathbf{x}_i - y_i \right)^2 \tag{3.109}$$

A closed-form solution of the LSE estimate of $\mathbf{A}$ can be obtained by taking the gradient of $e(\mathbf{A})$,

$$\nabla e(\mathbf{A}) = \sum_{i=1}^{n} 2(\mathbf{A}^t \mathbf{x}_i - y_i)\mathbf{x}_i = 2\mathbf{X}^t(\mathbf{XA} - \mathbf{Y}) \tag{3.110}$$

and equating it to zero. This yields the following equation:

$$\mathbf{X}^t\mathbf{XA} = \mathbf{X}^t\mathbf{Y} \tag{3.111}$$

Thus the LSE estimate $\mathbf{A}_{LSE}$ will be of the following form:

$$\mathbf{A}_{LSE} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.112}$$

$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ in Eq. (3.112) is also refereed to as the *pseudo-inverse* of $\mathbf{X}$ and is sometimes denoted as $\mathbf{X}^{\perp}$.

When $\mathbf{X}'\mathbf{X}$ is singular or some boundary conditions cause the LSE estimation in Eq. (3.112) to be unattainable, some numeric methods can be used to find an approximate solution. Instead of minimizing the quantity in Eq. (3.109), one can minimize the following quantity:

$$e(\mathbf{A}) = \| \mathbf{X}\mathbf{A} - \mathbf{Y} \|^2 + \alpha \| \mathbf{A} \|^2 \tag{3.113}$$

Following a similar procedure, one can obtain the LSE estimate to minimize the quantity above in the following form.

$$\mathbf{A}^{*}_{LSE} = (\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.114}$$

The LSE solution in Eq. (3.112) can be used for polynomial functions too. In the problem of polynomial curve fitting using the least square criterion, we are aiming to find the coefficients $\mathbf{A} = (a_0, a_1, a_2, \cdots, a_d)'$ that minimize the following quantity:

$$\min_{a_0, a_1, a_2, \cdots, a_d} E(Y - \hat{Y})^2 \tag{3.115}$$

where $\hat{Y} = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$

To obtain the LSE estimate of coefficients $\mathbf{A} = (a_0, a_1, a_2, \cdots, a_d)'$, simply change the formation of matrix $\mathbf{X}$ in Eq. (3.108) to the following:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^d \\ 1 & x_2 & \cdots & x_2^d \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^d \end{pmatrix} \tag{3.116}$$

Note that x_i^j in Eq. (3.108) means the j-th dimension of sample $\mathbf{x}_i$, while x_i^j in Eq. (3.116) means j-th order of value x_i. Therefore, the LSE estimate of polynomial coefficients $\mathbf{A}_{LSE} = (a_0, a_1, a_2, \cdots, a_d)'$ has the same form as Eq. (3.112).

3.2.1.3. MMSE/LSE for Nonlinear Functions

As the most general case, consider solving the following minimization problem:

$$\min_{g(\bullet) \in G_M} E\left[Y - g(X)\right]^2 \tag{3.117}$$

Since we need to deal with all possible nonlinear functions, taking a derivative does not work here. Instead, we use the property of conditional expectation to solve this minimization problem. By applying Eq. (3.38) to (3.117), we get

$$\begin{aligned} E_{X,Y}\left[Y-g(X)\right]^2 &= E_X\left\{E_{Y|X}\left[\left[Y-g(X)\right]^2 \mid X=x\right]\right\} \\ &= \int_{-\infty}^{\infty} E_{Y|X}\left[\left[Y-g(X)\right]^2 \mid X=x\right] f_X(x)dx \\ &= \int_{-\infty}^{\infty} E_{Y|X}\left[\left[Y-g(x)\right]^2 \mid X=x\right] f_X(x)dx \end{aligned} \tag{3.118}$$

Since the integrand is nonnegative in Eq. (3.118), the quantity in Eq. (3.117) will be minimized at the same time the following equation is minimized.

$$\min_{g(x)\in R} E_{Y|X}\left[\left[Y-g(x)\right]^2 \mid X=x\right] \tag{3.119}$$

Since $g(x)$ is a constant in the calculation of the conditional expectation above, the MMSE estimate can be obtained in the same way as the constant functions in Section 3.2.1.1. Thus, the MMSE estimate should take the following form:

$$\ddot{Y} = g_{MMSE}(X) = E_{Y|X}(Y \mid X) \tag{3.120}$$

If the value $X=x$ is observed and the value $E(Y \mid X=x)$ is used to predict Y, the mean squared error (MSE) is minimized and specified as follows:

$$E_{Y|X}\left[\left[Y-E_{Y|X}(Y \mid X=x)\right]^2 \mid X=x\right] = Var_{Y|X}(Y \mid X=x) \tag{3.121}$$

The overall MSE, averaged over all the possible values of X, is:

$$E_X\left[Y-E_{Y|X}(Y \mid X)\right]^2 = E_X\left\{E_{Y|X}\left[\left[Y-E_{Y|X}(Y \mid X)\right]^2 \mid X\right]\right\} = E_X\left[{}_{Y|X}Var(Y \mid X=x)\right] \tag{3.122}$$

It is important to distinguish between the overall MSE $E_X\left[Var_{Y|X}(Y \mid X)\right]$ and the MSE of the particular estimate when $X=x$, which is $Var_{Y|X}(Y \mid X=x)$. Before the value of X is observed, the expected MSE for the process of observing X and predicting Y is $E_X\left[Var_{Y|X}(Y \mid X)\right]$. On the other hand, after a particular value x of X has been observed and the prediction $E_{Y|X}(Y \mid X=x)$ has been made, the appropriate measure of MSE of the prediction is $Var_{Y|X}(Y \mid X=x)$.

In general, the form of the MMSE estimator for nonlinear functions depends on the form of the joint distribution of X and Y. There is no mathematical closed-form solution. To get the conditional expectation in Eq. (3.120), we have to perform the following integral:

$$\hat{Y}(x) = \int_{-\infty}^{\infty} y f_Y(y \mid X=x)dy \tag{3.123}$$

It is difficult to solve this integral calculation. First, different measures of x could determine different conditional pdf for the integral. Exact information about the pdf is often impossible to obtain. Second, there could be no analytic solution for the integral. Those difficulties reduce the interest of the MMSE estimation of nonlinear functions to theoretical aspects only. The same difficulties also exist for LSE estimation for nonlinear functions. Some certain classes of well-behaved nonlinear functions are typically assumed for LSE problems and numeric methods are used to obtain LSE estimate from sample data.

3.2.2. Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is the most widely used parametric estimation method, largely because of its efficiency. Suppose that a set of random samples $\mathbf{X}=\{X_1,X_2,\ldots,X_n\}$ is to be drawn independently according to a discrete or continuous distribution with the pf or the pdf $p(x|\Phi)$, where the parameter vector Φ belongs to some parameter space Ω. Given an observed vector $\mathbf{x}=(x_1,\cdots,x_n)$, the *likelihood* of the set of sample data vectors $\mathbf{x}$ with respect to Φ is defined as the joint pf or joint pdf $p_n(\mathbf{x}|\Phi)$; $p_n(\mathbf{x}|\Phi)$ is also referred to as the *likelihood function*.

MLE assumes the parameters of pdfs are fixed but unknown and aims to find the set of parameters that maximizes the likelihood of generating the observed data. For example, if the pdf $p_n(\mathbf{x}|\Phi)$ is assumed to be a Gaussian distribution $N(\boldsymbol{\mu},\boldsymbol{\Sigma})$, the components of Φ will then include exactly the components of mean-vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Since $X_1,X_2,\ldots,X_n$ are independent random variables, the likelihood can be rewritten as follows:

$$p_n(\mathbf{x}|\Phi)=\prod_{k=1}^{n} p(x_k|\Phi) \tag{3.124}$$

The likelihood $p_n(\mathbf{x}|\Phi)$ can be viewed as the probability of generating the sample data set $\mathbf{x}$ based on parameter set Φ. The *maximum likelihood estimator* of Φ is denoted as Φ_{MLE} that maximizes the likelihood $p_n(\mathbf{x}|\Phi)$. That is,

$$\Phi_{MLE}=\underset{\Phi}{\operatorname{argmax}}\, p_n(\mathbf{x}|\Phi) \tag{3.125}$$

This estimation method is called the *maximum likelihood estimation* method and is often abbreviated as MLE. Since the logarithm function is a monotonically increasing function, the parameter set Φ_{MLE} that maximizes the log-likelihood should also maximize the likelihood. If $p_n(\mathbf{x}|\Phi)$ is differentiable function of Φ, Φ_{MLE} can be attained by taking the partial derivative with respect to Φ and setting it to zero. Specifically, let Φ be a k-component parameter vector $\Phi=(\Phi_1,\Phi_2,\ldots,\Phi_k)^t$ and ∇_Φ be the gradient operator:

$$\nabla_{\Phi} = \begin{bmatrix} \dfrac{\partial}{\partial \Phi_1} \\ \vdots \\ \dfrac{\partial}{\partial \Phi_k} \end{bmatrix} \tag{3.126}$$

The log-likelihood becomes:

$$l(\Phi) = \log p_n(\mathbf{x} \mid \Phi) = \sum_{k=1}^{n} \log p(x_k \mid \Phi) \tag{3.127}$$

and its partial derivative is:

$$\nabla_{\Phi}\, l(\Phi) = \sum_{k=1}^{n} \nabla_{\Phi} \log p(x_k \mid \Phi) \tag{3.128}$$

Thus, the maximum likelihood estimate of Φ can be obtained by solving the following set of k equations:

$$\nabla_{\Phi}\, l(\Phi) = 0 \tag{3.129}$$

Example 3.1

Let's take a look at the maximum likelihood estimator of a univariate Gaussian pdf, given as the following equation:

$$p(x \mid \Phi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right] \tag{3.130}$$

where μ and σ^2 are the mean and the variance respectively. The parameter vector Φ denotes (μ, σ^2). The log-likelihood is:

$$\begin{aligned} \log p_n(\mathbf{x} \mid \Phi) &= \sum_{k=1}^{n} \log p(x_k \mid \Phi) \\ &= \sum_{k=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x_k-\mu)^2}{2\sigma^2} \right] \right) \\ &= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{k=1}^{n} (x_k - \mu)^2 \end{aligned} \tag{3.131}$$

and the partial derivative of the above expression is:

$$\frac{\partial}{\partial \mu}\log p_n(x|\Phi)=\sum_{k=1}^{n}\frac{1}{\sigma^2}(x_k-\mu)$$
$$\frac{\partial}{\partial \sigma^2}\log p_n(x|\Phi)=-\frac{n}{2\sigma^2}+\sum_{k=1}^{n}\frac{(x_k-\mu)^2}{2\sigma^4} \tag{3.132}$$

We set the two partial differential derivatives to zero,

$$\sum_{k=1}^{n}\frac{1}{\sigma^2}(x_k-\mu)=0$$
$$-\frac{n}{\sigma^2}+\sum_{k=1}^{n}\frac{(x_k-\mu)^2}{\sigma^4}=0 \tag{3.133}$$

The maximum likelihood estimates for μ and σ^2 are obtained by solving the above equations:

$$\mu_{MLE}=\frac{1}{n}\sum_{k=1}^{n}x_k=E(x)$$
$$\sigma^2_{MLE}=\frac{1}{n}\sum_{k=1}^{n}(x_k-\mu_{MLE})^2=E\left[(x-\mu_{MLE})^2\right] \tag{3.134}$$

Equation (3.134) indicates that the maximum likelihood estimation for mean and variance is just the sample mean and variance.

Example 3.2

For the multivariate Gaussian pdf $p(\mathbf{x})$

$$p(\mathbf{x}|\Phi)=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{3.135}$$

The maximum likelihood estimates of $\boldsymbol{\mu}$ and Σ can be obtained by a similar procedure.

$$\hat{\boldsymbol{\mu}}_{MLE}=\frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$
$$\hat{\Sigma}_{MLE}=\frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})'=E\left[(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})'\right] \tag{3.136}$$

Once again, the maximum likelihood estimation for mean vector and covariance matrix is the sample mean vector and sample covariance matrix.

In some situations, maximum likelihood estimation of Φ may not exist, or the maximum likelihood estimator may not be uniquely defined, i.e., there may be more than one MLE of Φ for a specific set of sample values. Fortunately, according to Fisher's theorem, for most practical problems with a well-behaved family of distributions, the MLE exists and is uniquely defined [4, 25, 26].

In fact, the maximum likelihood estimator can be proven to be sound under certain conditions. As mentioned before, the estimator $\theta(\mathbf{X})$ is a function of the vector of random variables $\mathbf{X}$ that represent the sample data. $\theta(\mathbf{X})$ itself is also a random variable, with a distribution determined by joint distributions of $\mathbf{X}$. Let $\tilde{\Phi}$ be the parameter vector of true distribution $p(x|\Phi)$ from which the samples are drawn. If the following three conditions hold:

1. The sample $\mathbf{x}$ is a drawn from the assumed family of distribution,
2. The family of distributions is well behaved,
3. The sample $\mathbf{x}$ is large enough,

then maximum likelihood estimator, Φ_{MLE}, has a Gaussian distribution with a mean $\tilde{\Phi}$ and a variance of the form $1/nB_x^2$ [26], where n is the size of sample and B_x is the *Fisher information*, which is determined solely by $\tilde{\Phi}$ and $\mathbf{x}$. An estimator is said to be *consistent*, iff the estimate will converge to the true distribution when there is infinite number of training samples.

$$\lim_{n\to\infty} \Phi_{MLE} = \tilde{\Phi} \tag{3.137}$$

Φ_{MLE} is a consistent estimator based on the analysis above. In addition, it can be shown that no consistent estimator has a lower variance than Φ_{MLE}. In other words, no estimator provides a closer estimate of the true parameters than the maximum likelihood estimator.

3.2.3. Bayesian Estimation and MAP Estimation

Bayesian estimation has a different philosophy than maximum likelihood estimation. While MLE assumes that the parameter Φ[3] is fixed but unknown, Bayesian estimation assumes that the parameter Φ itself is a random variable with a prior distribution $p(\Phi)$. Suppose we observe a sequence of random samples $\mathbf{x}=\{x_1, x_2, \ldots, x_n\}$, which are i.i.d. with a pdf $p(x|\Phi)$. According to Bayes' rule, we have the posterior distribution of Φ as:

$$p(\Phi|\mathbf{x}) = \frac{p(\mathbf{x}|\Phi)p(\Phi)}{p(\mathbf{x})} \propto p(\mathbf{x}|\Phi)p(\Phi) \tag{3.138}$$

[3] For simplicity, we assume the parameter Φ is a scalar instead of a vector here. However, the extension to a parameter vector Φ can be derived according to a similar procedure.

In Eq. (3.138), we dropped the denominator $p(\mathbf{x})$ here because it is independent of the parameter Φ. The distribution in Eq. (3.138) is called the posterior distribution of Φ because it is the distribution of Φ after we observed the values of random variables $X_1, X_2, \ldots, X_n$.

3.2.3.1. Prior and Posterior Distributions

For mathematical tractability, conjugate priors are often used in Bayesian estimation. Suppose a random sample is taken of a known distribution with pdf $p(\mathbf{x}|\Phi)$. A conjugate prior for the random variable (or vector) is defined as the prior distribution for the parameters of the probability density function of the random variable (or vector), such that the class-conditional pdf $p(\mathbf{x}|\Phi)$, the posterior distribution $p(\Phi|\mathbf{x})$, and the prior distribution $p(\Phi)$ belong to the same distribution family. For example, it is well known that the conjugate prior for the mean of a Gaussian pdf is also a Gaussian pdf [4]. Now, let's derive such a posterior distribution $p(\Phi|\mathbf{x})$ from the widely used Gaussian conjugate prior.

Example 3.3

Suppose $X_1, X_2, \ldots, X_n$ are drawn from a Gaussian distribution for which the mean Φ is a random variable and the variance σ^2 is known. The likelihood function $p(\mathbf{x}|\Phi)$ can be written as:

$$p(\mathbf{x}|\Phi) = \frac{1}{(2\pi)^{n/2}\sigma^n}\exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i-\Phi}{\sigma}\right)^2\right] \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i-\Phi}{\sigma}\right)^2\right] \tag{3.139}$$

To further simply Eq. (3.139), we could use Eq. (3.140)

$$\sum_{i=1}^{n}(x_i-\Phi)^2 = n(\Phi-\bar{x}_n)^2 + \sum_{i=1}^{n}(x_i-\bar{x}_n)^2 \tag{3.140}$$

where $\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n}x_i =$ the sample mean of $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$.

Let's rewrite $p(\mathbf{x}|\Phi)$ in Eq. (3.139) into Eq. (3.141):

$$p(\mathbf{x}|\Phi) \propto \exp\left[-\frac{n}{2\sigma^2}(\Phi-\bar{x}_n)^2\right]\exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\bar{x}_n)^2\right] \tag{3.141}$$

Now suppose the prior distribution of Φ is also a Gaussian distribution with mean μ and variance ν^2, i.e., the prior distribution $p(\Phi)$ is given as follows:

$$p(\Phi) = \frac{1}{(2\pi)^{1/2}\nu}\exp\left[-\frac{1}{2}\left(\frac{\Phi-\mu}{\nu}\right)^2\right] \propto \exp\left[-\frac{1}{2}\left(\frac{\Phi-\mu}{\nu}\right)^2\right] \tag{3.142}$$

By combining Eqs. (3.141) and (3.142) while dropping the second term in Eq. (3.141) we could attain the posterior pdf $p(\Phi \mid \mathbf{x})$ in the following equation:

$$p(\Phi \mid \mathbf{x}) \propto \exp\left\{-\frac{1}{2}\left[\frac{n}{\sigma^2}(\Phi - \bar{x}_n)^2 + \frac{1}{\nu^2}(\Phi - \mu)^2\right]\right\} \tag{3.143}$$

Now if we define ρ and τ as follows:

$$\rho = \frac{\sigma^2\mu + n\nu^2\bar{x}_n}{\sigma^2 + n\nu^2} \tag{3.144}$$

$$\tau^2 = \frac{\sigma^2\nu^2}{\sigma^2 + n\nu^2} \tag{3.145}$$

We can rewrite Eq. (3.143) as:

$$p(\Phi \mid \mathbf{x}) \propto \exp\left\{-\frac{1}{2}\left[\frac{1}{\tau^2}(\Phi - \rho)^2 + \frac{n}{\sigma^2 + n\nu^2}(\bar{x}_n - \mu)^2\right]\right\} \tag{3.146}$$

Since the second term in Eq. (3.146) does not depend on Φ, it can be absorbed in the constant factor. Finally, we have the posterior pdf in the following form:

$$p(\Phi \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi}\tau}\exp\left[\frac{-1}{2\tau^2}(\Phi - \rho)^2\right] \tag{3.147}$$

Equation (3.147) shows that the posterior pdf $p(\Phi \mid \mathbf{x})$ is a Gaussian distribution with mean ρ and variance τ^2 as defined in Eqs. (3.144) and (3.145). The Gaussian prior distribution defined in Eq. (3.142) is a conjugate prior.

3.2.3.2. General Bayesian Estimation

The foremost requirement of a good estimator θ is that it can yield an estimate of Φ ($\theta(\mathbf{X})$) which is close to the real value Φ. In other words, a good estimator is one for which it is highly probable that the error $\theta(\mathbf{X}) - \Phi$ is close to 0. In general, we can define a loss function[4] $R(\Phi, \bar{\Phi})$. It measures the loss or cost associated with the fact that the true value of the parameter is Φ while the estimate is $\bar{\Phi}$. When only the prior distribution

[4] Bayesian estimation and loss functions are based on Bayes' decision theory, described in Chapter 4.

$p(\Phi)$ is available and no sample data has been observed, if we choose one particular estimate $\bar{\Phi}$, the expected loss is:

$$E\left[R(\Phi,\bar{\Phi})\right]=\int R(\Phi,\bar{\Phi})p(\Phi)d\Phi \tag{3.148}$$

The fact that we could derive posterior distribution from the likelihood function and the prior distribution [as shown in the derivation of Eq. (3.147)] is very important here because it allows us to compute the expected posterior loss after sample vector $\mathbf{x}$ is observed. The expected posterior loss associated with estimate $\bar{\Phi}$ is:

$$E\left[R(\Phi,\bar{\Phi})\mid\mathbf{x}\right]=\int R(\Phi,\bar{\Phi})p(\Phi\mid\mathbf{x})d\Phi \tag{3.149}$$

The Bayesian estimator of Φ is defined as the estimator that attains minimum Bayes risk, that is, minimizes the expected posterior loss function (3.149). Formally, the Bayesian estimator is chosen according to:

$$\theta_{Bayes}(\mathbf{x})=\underset{\hat{\theta}}{\operatorname{argmin}}\ E\left[R(\Phi,\theta(\mathbf{x}))\mid\mathbf{x}\right] \tag{3.150}$$

The Bayesian estimator of Φ is the estimator θ_{Bayes} for which Eq. (3.150) is satisfied for every possible value of $\mathbf{x}$ of random vector $\mathbf{X}$. Therefore, the form of the Bayesian estimator θ_{Bayes} should depend only on the loss function and the prior distribution, but not the sample value $\mathbf{x}$.

One of the most common loss functions used in statistical estimation is the mean squared error function [20]. The mean squared error function for Bayesian estimation should have the following form:

$$R(\Phi,\theta(\mathbf{x}))=(\Phi-\theta(\mathbf{x}))^2 \tag{3.151}$$

In order to find the Bayesian estimator, we are seeking θ_{Bayes} to minimize the expected posterior loss function:

$$E\left[R(\Phi,\theta(\mathbf{x}))\mid\mathbf{x}\right]=E\left[(\Phi-\theta(\mathbf{x}))^2\mid\mathbf{x}\right]=E(\Phi^2\mid\mathbf{x})-2\theta(\mathbf{x})E(\Phi\mid\mathbf{x})-\theta(\mathbf{x})^2 \tag{3.152}$$

The minimum value of this function can be obtained by taking the partial derivative of Eq. (3.152) with respect to $\theta(\mathbf{x})$. Since the above equation is simply a quadratic function of $\theta(\mathbf{x})$, it can be shown that the minimum loss can be achieved when θ_{Bayes} is chosen based on the following equation:

$$\theta_{Bayes}(\mathbf{x})=E(\Phi\mid\mathbf{x}) \tag{3.153}$$

Equation (3.153) translates into the fact that the Bayesian estimate of the parameter Φ for mean squared error function is equal to the mean of the posterior distribution of Φ. In the following section, we discuss another popular loss function (MAP estimation) that also generates the same estimate for certain distribution functions.

3.2.3.3. MAP Estimation

One intuitive interpretation of Eq. (3.138) is that a prior pdf $p(\Phi)$ represents the relative likelihood before the values of $X_1, X_2, \ldots, X_n$ have been observed; while the posterior pdf $p(\Phi \mid \mathbf{x})$ represents the relative likelihood after the values of $X_1, X_2, \ldots, X_n$ have been observed. Therefore, choosing an estimate $\bar{\Phi}$ that maximizes the posterior probability is consistent without intuition. This estimator is in fact the *maximum posterior probability* (MAP) estimator and is the most popular Bayesian estimator.

The loss function associated with the MAP estimator is the so-called uniform loss function [20]:

$$R(\Phi, \theta(\mathbf{x})) = \begin{cases} 0, & \text{if } |\theta(\mathbf{x}) - \Phi| \le \Delta \\ 1, & \text{if } |\theta(\mathbf{x}) - \Phi| > \Delta \end{cases} \quad \text{where } \Delta > 0 \tag{3.154}$$

Now let's see how this uniform loss function results in MAP estimation. Based on the loss function defined above, the expected posterior loss function is:

$$\begin{aligned} E(R(\Phi, \theta(\mathbf{x})) \mid \mathbf{x}) &= P(|\theta(\mathbf{x}) - \Phi| > \Delta \mid \mathbf{x}) \\ &= 1 - P(|\theta(\mathbf{x}) - \Phi| \le \Delta \mid \mathbf{x}) = 1 - \int_{\theta(\mathbf{x})-\Delta}^{\theta(\mathbf{x})+\Delta} p(\Phi \mid \mathbf{x}) \end{aligned} \tag{3.155}$$

The quantity in Eq. (3.155) is minimized by maximizing the shaded area under $p(\Phi \mid \mathbf{x})$ over the interval $[\theta(\mathbf{x}) - \Delta, \theta(\mathbf{x}) + \Delta]$ in Figure 3.16. If $p(\Phi \mid \mathbf{x})$ is a smooth curve and Δ is small enough, the shaded area can be computed roughly as:

$$\int_{\theta(\mathbf{x})-\Delta}^{\theta(\mathbf{x})+\Delta} p(\Phi \mid \mathbf{x}) \cong 2\Delta p(\Phi \mid \mathbf{x})\Big|_{\Phi=\theta(\mathbf{x})} \tag{3.156}$$

Thus, the shaded area can be approximately maximized by choosing $\theta(\mathbf{x})$ to be the maximum point of $p(\Phi \mid \mathbf{x})$. This concludes our proof the using the error function in Eq. (3.154) indeed will generate MAP estimator.

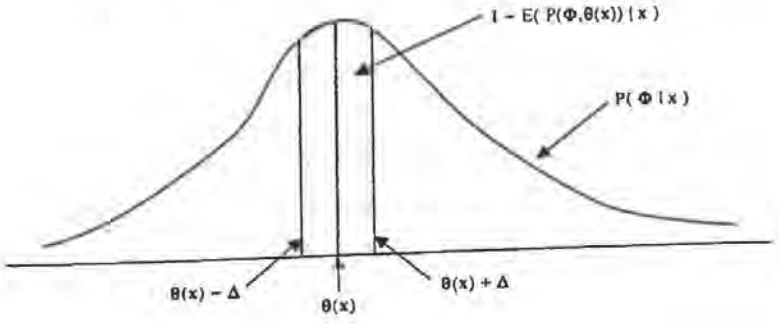


Figure 3.16 Illustration of the minimum expected posterior loss function for MAP estimation [20].

MAP estimation is to find the parameter estimate Φ_{MAP} or estimator $\theta_{MAP}(\mathbf{x})$ that maximizes the posterior probability,

$$\Phi_{MAP} = \theta_{MAP}(\mathbf{x}) = \underset{\Phi}{\operatorname{argmax}}\ p(\Phi \mid \mathbf{x}) = \underset{\Phi}{\operatorname{argmax}}\ p(\mathbf{x} \mid \Phi) p(\Phi) \tag{3.157}$$

Φ_{MAP} can also be specified in the logarithm form as follows:

$$\Phi_{MAP} = \underset{\Phi}{\operatorname{argmax}}\ \left[\log p(\mathbf{x} \mid \Phi) + \log p(\Phi)\right] \tag{3.158}$$

Φ_{MAP} can be attained by solving the following partial differential equation:

$$\frac{\partial \log p(\mathbf{x} \mid \Phi)}{\partial \Phi} + \frac{\partial \log p(\Phi)}{\partial \Phi} = 0 \tag{3.159}$$

Thus the MAP equation for finding Φ_{MAP} can be established.

$$\left.\frac{\partial \log p(\mathbf{x} \mid \Phi)}{\partial \Phi}\right|_{\Phi=\Phi_{MAP}} = \left.\frac{-\partial \log p(\Phi)}{\partial \Phi}\right|_{\Phi=\Phi_{MAP}} \tag{3.160}$$

There are interesting relationships between MAP estimation and MLE estimation. The prior distribution is viewed as the knowledge of the statistics of the parameters of interest before any sample data is observed. For the case of MLE, the parameter is assumed to be fixed but unknown. That is, there is no preference (knowledge) of what the values of parameters should be. The prior distribution $p(\Phi)$ can only be set to constant for the entire parameter space, and this type of prior information is often referred to as *non-informative prior* or *uniform prior*. By substituting $p(\Phi)$ with a uniform distribution in Eq. (3.157), MAP estimation is identical to MLE. In this case, the parameter estimation is solely determined by the observed data. A sufficient amount of training data is often a requirement for MLE. On the other hand, when the size of the training data is limited, the use of the prior density becomes valuable. If some prior knowledge of the distribution of the parameters can be obtained, MAP estimation provides a way of incorporating prior information in the parameter learning process.

Example 3.4

Now, let's formulate MAP estimation for Gaussian densities. As described in Section 3.2.3.1, the conjugate prior distribution for a Gaussian density is also a Gaussian distribution. Similarly, we assumed random variables $X_1, X_2, \ldots, X_n$ drawn from a Gaussian distribution for which the mean Φ is unknown and the variance σ^2 is known, while the conjugate prior distribution of Φ is a Gaussian distribution with mean μ and variance ν^2. It is shown in Section 3.2.3.1 that the posterior pdf can be formulated as in Eq. (3.147). The MAP estimation for Φ can be solved by taking the derivative of Eq. (3.147) with respect to Φ:

$$\Phi_{MAP} = \rho = \frac{\sigma^2 \mu + n\nu^2 \bar{x}_n}{\sigma^2 + n\nu^2} \tag{3.161}$$

where n is the total number of training samples and $\bar{x}_n$ the sample mean.

The MAP estimate of the mean Φ is a weighted average of the sample mean $\bar{x}_n$ and the prior mean. When n is zero (when there is no training data at all), the MAP estimate is simply the prior mean μ. On the other hand, when n is large ($n \to \infty$), the MAP estimate will converge to the maximum likelihood estimate. This phenomenon is consistent with our intuition and is often referred to as *asymptotic equivalence* or *asymptotic convergence*. Therefore, in practice, the difference between MAP estimation and MLE is often insignificant when a large amount of training data is available. When the prior variance ν^2 is very large (e.g., $\nu^2 >> \sigma^2 / n$), the MAP estimate will converge to the ML estimate because a very large ν^2 translates into a non-informative prior.

It is important to note that the requirement of learning prior distributions for MAP estimation is critical. In some cases, the prior distribution is very difficult to estimate and MLE is still an attractive estimation method. As mentioned before, the MAP estimation framework is particularly useful for dealing with sparse data, such as parameter adaptation. For example, in speaker adaptation, the speaker-independent (or multiple speakers) database can be used to first estimate the prior distribution [9]. The model parameters are adapted to a target speaker through a MAP framework by using limited speaker-specific training data as discussed in Chapter 9.

3.3. SIGNIFICANCE TESTING

Significance testing is one of the most important theories and methods of statistical *inference*. A problem of statistical inference, or, more simply, a statistics problem, is one in which data that have been generated in accordance with some unknown probability distribution must be analyzed, and some type of inference about the unknown distribution must be made. Hundreds of test procedures have developed in statistics for various kinds of hypotheses testing. We focus only on tests that are used in spoken language systems.

The selection of appropriate models for the data or systems is essential for spoken language systems. When the distribution of certain sample data is unknown, it is usually appropriate to make some assumptions about the distribution of the data with a distribution function whose properties are well known. For example, people often use Gaussian distributions to model the distribution of background noise in spoken language systems. One important issue is how good our assumptions are, and what the appropriate values of the parameters for the distributions are, even when we can use the methods in Section 3.2 to estimate parameters from sample data. Statistical tests are often applied to determine if the distribution with specific parameters is appropriate to model the sample data. In this section, we describe the most popular testing method for the goodness of distribution fitting – the χ^2 goodness-of-fit test.

Another important type of statistical tests is designed to evaluate the excellence of two different methods or algorithms for the same tasks when there is uncertainty regarding the results. To assure that the two systems are evaluated on the same or similar conditions, experimenters often carefully choose similar or even the exactly same data sets for testing. This is why we refer to this type of statistical test as a *paired observations* test. In both speech recognition and speech synthesis, the paired observations test is a very important tool for interpreting the comparison results.

3.3.1. Level of Significance

We now consider statistical problems involving a parameter ϕ whose value is unknown but must lie in a certain parameter space Ω. In statistical tests, we let H_0 denote the hypothesis that $\phi \in \Omega_0$ and let H_1 denote the hypothesis that $\phi \in \Omega_1$. The subsets Ω_0 and Ω_1 are disjoint and $\Omega_0 \cup \Omega_1 = \Omega$, so exactly one of the hypotheses H_0 and H_1 must be true. We must now decide whether to accept H_0 or H_1 by observing a random sample $X_1, \cdots, X_n$ drawn from a distribution involving the unknown parameter ϕ. A problem like this is called hypotheses testing. A procedure for deciding whether to accept H_0 or H_1 is called a *test procedure* or simply a *test*. The hypothesis H_0 is often referred to as the *null hypothesis* and the hypothesis H_1 as the *alternative hypothesis*. Since there are only two possible decisions, accepting H_0 is equivalent to rejecting H_1 and rejecting H_0 is equivalent to accepting H_1. Therefore, in testing hypotheses, we often use the terms *accepting or rejecting the null hypothesis* H_0 as the only decision choices.

Usually we are presented with a random sample $\mathbf{X} = (X_1, \cdots, X_n)$ to help us in making the test decision. Let S denote the sample space of n-dimensional random vector $\mathbf{X}$. The testing procedure is equivalent to partitioning the sample space S into two subsets. One subset specifies the values of $\mathbf{X}$ for which one will accept H_0 and the other subset specifies the values of $\mathbf{X}$ for which one will reject H_0. The second subset is called the *critical region* and is often denoted as C.

Since there is uncertainty associated with the test decision, for each value of $\phi \in \Omega$, we are interested in the probability $\rho(\phi)$ that the testing procedure rejects H_0. The function $\rho(\phi)$ is called the *power function* of the test and can be specified as follows:

$$\rho(\phi) = P(\mathbf{X} \in C \mid \phi) \tag{3.162}$$

For $\phi \in \Omega_0$, the decision to reject H_0 is incorrect. Therefore, if $\phi \in \Omega_0$, $\rho(\phi)$ is the probability that the statistician will make an incorrect decision (false rejection). In statistical tests, an upper bound α_0 $(0 < \alpha_0 < 1)$ is specified, and we only consider tests for which $\rho(\phi) \le \alpha_0$ for every value of $\phi \in \Omega_0$. The upper bound α_0 is called the *level of significance*. The smaller α_0 is, the less likely it is that the test procedure will reject H_0. Since α_0 specifies the upper bound for false rejection, once a hypothesis is rejected by the test procedure, we can be $(1-\alpha_0)$ confident the decision is correct. In most applications, α_0 is set to be 0.05 and the test is said to be carried out at the 0.05 level of significance or 0.95 level of confidence.

We define the size α of a given test as the maximum probability, among all the values of ϕ which satisfy the null hypothesis, of making an incorrect decision.

$$\alpha = \max_{\theta \in \Omega_0} \rho(\phi) \tag{3.163}$$

Once we obtain the value of α, the test procedure is straightforward. First, the statistician specifies a certain level of significance α_0 in a given problem of testing hypotheses, then he or she rejects the null hypothesis if the size α is such that $\alpha \leq \alpha_0$.

The size α of a given test is also called the *tail area* or the *p-value* corresponding to the observed value of data sample **X** because it corresponds to tail area of the distribution. The hypothesis will be rejected if the level of significance α_0 is such that $\alpha_0 > \alpha$ and should be accepted for any value of $\alpha_0 < \alpha$. Alternatively, we can say the observed value of **X** is *just significant* at the level of significance α without using the level of significance α_0. Therefore, if we had found that the observed value of one data sample **X** was just significant at the level of 0.0001, while the other observed value of data sample **Y** was just significant at the level of 0.001, then we can conclude the sample **X** provides much stronger evidence against H_0. In statistics, an observed value of one data sample **X** is generally said to be statistically significant if the corresponding tail area is smaller than the traditional value 0.05. For cases requiring more significance (confidence), 0.01 can be used.

A statistically significant observed data sample **X** that provides strong evidence against H_0 does not necessary provide strong evidence that the actual value of ϕ is significantly far away from parameter set Ω_0. This situation can arise, particularly when the size of random data sample is large, because a test with larger sample size will in general reject hypotheses with more confidence, unless the hypothesis is indeed the true one.

3.3.2. Normal Test (Z-Test)

Suppose we need to find whether a coin toss is fair or not. Let p be the probability of heads. The hypotheses to be tested are as follows:

$H_0 : p = 1/2$

$H_1 : p \neq 1/2$

We assume that a random sample size n is taken, and let random variable M denote the number of times we observe heads as the result. The random variable M has a binomial distribution $B(n, 1/2)$. Because of the shape of binomial distribution, M can lie on either side of the mean. This is why it is called a typical *two-tailed test*. The tail area or p-value for the observed value k can be computed as:

$$p = \begin{cases} 2P(k \leq M \leq n) & \text{for } k > n/2 \\ 2P(0 \leq M \leq k) & \text{for } k < n/2 \\ 1.0 & \text{for } k = n/2 \end{cases} \tag{3.164}$$

The p-value in Eq. (3.164) can be computed directly using the binomial distribution. The test procedure will reject H_0 when p is less than the significance level α_0.

In many situations, the p-value for the distribution of data sample $\mathbf{X}$ is difficult to obtain due to the complexity of the distribution. Fortunately, if some statistic Z of the data sample $\mathbf{X}$ has some well-known distribution, the test can then be done in the Z domain instead. If n is large enough ($n > 50$), a *normal test* (or *Z-test*) can be used to approximate a binomial probability. Under H_0, the mean and variance for M are $E(M) = n/2$ and $Var(M) = n/4$. The new random variable Z is defined as,

$$Z = \frac{|M - n/2| - 1/2}{\sqrt{n/4}} \tag{3.165}$$

which can be approximated as standard Gaussian distribution $N(0,1)$ under H_0. The p-value can now be computed as $p = 2P(Z \geq z)$ where z is the realized value of Z after M is observed. Thus, H_0 is rejected if $p < \alpha_0$, where α_0 is the level of significance.

3.3.3. χ^2 Goodness-of-Fit Test

The normal test (Z-test) can be extended to test the hypothesis that a given set of data came from a certain distribution with all parameters specified. First let's look at the case of discrete distribution fitting.

Suppose that a large population consists of items of k different types and let p_i be the probability that a random selected item belongs to type i. Now, let $q_1,\ldots,q_k$ be a set of specific numbers satisfying the probabilistic constraint ($q_i \geq 0$ for $i = 1,\ldots,k$ and $\sum_{i=1}^{k} q_i = 1$). Finally, suppose that the following hypotheses are to be tested:

$H_0 : p_i = q_i$ for $i = 1,\ldots,k$

$H_1 : p_i \neq q_i$ for at least one value of i

Assume that a random sample of size n is to be taken from the given population. For $i = 1,\ldots,k$, let N_i denote the number of observations in the random sample which are of type i. Here, $N_1,\ldots,N_k$ are nonnegative numbers and $\sum_{i=1}^{k} N_i = n$. Random variables $N_1,\ldots,N_k$ have a multinomial distribution. Since the p-value for the multinomial distribution is hard to obtain, instead we use another statistic about $N_1,\ldots,N_k$. When H_0 is true, the expected number of observations of type i is nq_i. In other words, the difference between the actual number of observations N_i and the expected number nq_i should be small when H_0 is true. It seems reasonable to base the test on the differences $N_i - nq_i$ and to reject H_0 when the differences are large. It can be proved [14] that the following random variable λ

$$\lambda = \sum_{i=1}^{k} \frac{(N_i - nq_i)^2}{nq_i} \tag{3.166}$$

converges to the χ^2 distribution with $k-1$ degrees of freedom as the sample size $n \to \infty$.

A χ^2 test of goodness-of-fit can be carried out in the following way. Once a level of significance α_0 is specified, we can use the following p-value function to find critical point c:[5]

$$P(\lambda > c) = 1 - F_{\chi^2}(x = c) \simeq \alpha_0 \tag{3.167}$$

where $F_{\chi^2}(x)$ is the distribution function for χ^2 distribution. The test procedure simply rejects H_0 when the realized value λ is such that $\lambda > c$. Empirical results show that the χ^2 distribution will be a good approximation to the actual distribution of λ as long as the value of each expectation nq_i is not too small (≥ 5). The approximation should still be satisfactory if $nq_i \geq 1.5$ for $i = 1,\ldots,k$.

For continuous distributions, a modified χ^2 goodness-of-fit test procedure can be applied. Suppose that we would like to hypothesize a null hypothesis H_0 in which continuous random sample data $X_1,\ldots,X_k$ are drawn from a certain continuous distribution with all parameters specified or estimated. Also, suppose the observed values of random sample $x_1,\ldots,x_k$ are bounded within interval Ω. First, we divide the range of the hypothesized distribution into m subintervals within interval Ω such that the expected number of values, say E_i, in each interval is at least 5. For $i = 1,\ldots,k$, we let N_i denote the number of observations in the i^{th} subintervals. As in Eq. (3.166), one can prove that the following random variable λ

$$\lambda = \sum_{i=1}^{m} \frac{(N_i - E_i)^2}{E_i} \tag{3.168}$$

converges to the χ^2 distribution with $m-k-1$ degrees of freedom as the sample size $n \to \infty$, where k is the number of parameters that must be estimated from the sample data in order to calculate the expected number of values, E_i. Once the χ^2 distribution is established, the same procedure can be used to find the critical c in Eq. (3.167) to make test decisions.

Example 3.5

Suppose we are given a random variable X of sample size 100 points and we want to determine whether we can reject the following hypothesis:

$$H_0 : X \sim N(0,1) \tag{3.169}$$

To perform χ^2 goodness-of-fit test, we first divide the range of X into 10 subintervals. The corresponding probability falling in each subinterval, the expected number of points falling in each subinterval and the actual number of points falling in each subintervals [10] are illustrated in Table 3.1.

[5] Since χ^2 pdf is a monotonic function, the test is a one-tail test. Thus, we only need to calculate one tail area.

Table 3.1 The probability falling in each subinterval of an $N(0,1)$, and 100 sample points, the expected number of points falling in each subinterval, and the actual number of points falling in each subinterval [10].

Subinterval I_i	$P(X \in I_i)$	$E_i = 100P(X \in I_i)$	N_i
[-∞, -1.6]	0.0548	5.48	2
[-1.6, -1.2]	0.0603	6.03	9
[-1.2, -0.8]	0.0968	9.68	6
[-0.8, -0.4]	0.1327	13.27	11
[-0.4, 0.0]	0.1554	15.54	19
[0.0, 0.4]	0.1554	15.54	25
[0.4, 0.8]	0.1327	13.27	17
[0.8, 1.2]	0.0968	9.68	2
[1.2, 1.6]	0.0603	6.03	6
[-1.6, ∞]	0.0548	5.48	3

The value for λ can then be calculated as follows:

$$\lambda = \sum_{i=1}^{m} \frac{(N_i - E_i)^2}{E_i} = 18.286$$

Since λ can be approximated as a χ^2 distribution with $m-k-1=10-0-1=9$ degrees of freedom, the critical point c at the 0.05 level of significance is calculated[6] to be 16.919 according to Eq. (3.167). Thus, we should reject the hypothesis H_0 because the calculated λ is greater than the critical point c.

The χ^2 goodness-of-fit test at the 0.05 significance level is in general used to determine when a hypothesized distribution is not an adequate distribution to use. To accept the diswtribution as a good fit, one needs to make sure the hypothesized distribution cannot be rejected at the 0.4 to 0.5 level-of-significance. The alternative is to use the χ^2 goodness-of-fit test for a number of potential distributions and select the one with smallest calculated χ^2.

When all the parameters are specified (instead of estimated), the Kolmogorov-Smirnov test [5] can also be used for the goodness-of-fit test. The Kolmogorov-Smirnov test in general is a more powerful test procedure when the sample size is relatively small.

3.3.4. Matched-Pairs Test

In this section, we discuss experiments in which two different methods (or systems) are to be compared to learn which one is better. To assure the two methods are evaluated under

[6] In general, we use a cumulative distribution function table to find the point with specific desired cumulative probability for complicated distributions, like the χ^2 distribution.

similar conditions, two closely resemble data samples or ideally the same data sample should be used to evaluate both methods. This type of hypotheses test is called *matched-paired* test [5].

3.3.4.1. The Sign Test

For $i = 1,\ldots,n$, let p_i denote the probability that method A is better than method B when testing on the i^{th} paired data sample. We shall assume that the probability p_i has the same value p for each of the n pairs. Suppose we wish to test the null hypothesis that method A is no better than method B. That is, the hypotheses to be tested have the following form:

$H_0 : p \leq 1/2$

$H_1 : p > 1/2$

Suppose that, for each pair of data samples, either one method or the other will appear to be better, and the two methods cannot tie. Under these assumptions, the n pairs represent n Bernoulli trials, for each of which there is probability p that method A yields better performance. Thus the number of pairs M in which method A yields better performance will have a binomial distribution $B(n, p)$. For the simple sign test where one needs to decide which method is better, p will be set to $1/2$. Hence a reasonable procedure is to reject H_0 if $M > c$, where c is a critical point. This procedure is called a signed test. The critical point can be found according to

$$P(M > c) = 1 - F_B(x = c) = \alpha_0 \tag{3.170}$$

where $F_B(x)$ is the distribution for binomial distribution. Thus, for observed values $M > c$, we will reject H_0.

3.3.4.2. Magnitude-Difference Test

The only information that the sign test utilizes from each pair of data samples, is the sign of the difference between two performances. To do a sign test, one does not need to obtain a numeric measurement of the magnitude of the difference between the two performances. However, if the measurement of magnitude of the difference for each pair is available, a test procedure based on the relative magnitudes of the differences can be used [11].

We assume now that the performance of each method can be measured for any data samples. For $i = 1,\ldots,n$, let A_i denote the performance of the method A on the i^{th} pair of data samples and B_i denote the performance of the method B on the i^{th} pair of data sample. Moreover, we shall let $D_i = A_i - B_i$. Since $D_1,\ldots,D_n$ are generated on n different pairs of data samples, they should be independent random variables. We also assume that $D_1,\ldots,D_n$ have the same distribution. Suppose now we are interested in testing the null hypothesis that method A and method B have on the average the same performance on the n pairs of data samples.

Let μ_D be the mean of D_i. The MLE estimate of μ_D is:

$$\mu_D = \sum_{i=1}^{n} \frac{D_i}{n} \tag{3.171}$$

The test hypotheses are:

$H_0 : \mu_D = 0$

$H_1 : \mu_D \neq 0$

The MLE estimate of the variance of D_i is

$$\sigma_D^2 = \frac{1}{n} \sum_{i=1}^{n} (D_i - \mu_D)^2 \tag{3.172}$$

We define a new random variable Z as follows:

$$Z = \frac{\mu_D}{\sigma_D / \sqrt{n}} \tag{3.173}$$

If n is large enough (> 50), Z is proved to have a standard Gaussian distribution $N(0,1)$. The normal test procedure described in Section 3.3.2 can be used to test H_0. This type of matched-paired tests usually depends on having enough pairs of data samples for the assumption that Z can be approximated with a Gaussian distribution. It also requires enough data samples to estimate the mean and variance of the D_i.

3.4. INFORMATION THEORY

Transmission of information is a general definition of what we call communication. Claude Shannon's classic paper of 1948 gave birth to a new field in information theory that has become the cornerstone for coding and digital communication. In the paper titled "A Mathematical Theory of Communication," he wrote [21]:

> *The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.*

Information theory is a mathematical framework for approaching a large class of problems related to encoding, transmission, and decoding information in a systematic and disciplined way. Since speech is a form of communication, information theory has served as the underlying mathematical foundation for spoken language processing.

3.4.1. Entropy

Three interpretations can be used to describe the quantity of *information*: (1) the amount of uncertainty before seeing an event, (2) the amount of surprise when seeing an event, and (3)

the amount of information after seeing an event. Although these three interpretations seem slightly different, they are virtually the same under the framework of information theory.

According to information theory, the information derivable from outcome x_i depends on its probability. If the probability $P(x_i)$ is small, we can derive a large degree of information, because the outcome that it has occurred is very rare. On the other hand, if the probability is large, the information derived will be small, because the outcome is well expected. Thus, the amount of information is defined as follows:

$$I(x_i) = \log \frac{1}{P(x_i)} \tag{3.174}$$

The reason to use a logarithm can be interpreted as follows. The information for two independent events to occur (where the joint probability is the multiplication of both individual probabilities) can be simply carried out by the addition of the individual information of each event. When the logarithm base is 2, the unit of information is called a *bit*. This means that one bit of information is required to specify the outcome. In this probabilistic framework, the amount of information represents uncertainty. Suppose X is a discrete random variable taking value x_i (referred to as a symbol) from a finite or countable infinite sample space $S = \{x_1, x_2, \ldots, x_i, \ldots\}$ (referred to as an alphabet). The symbol x_i is produced from an information source with alphabet S, according to the probability distribution of the random variable X. One of the most important properties of an information source is the entropy $H(S)$ of the random variable X, defined as the average amount of information (expected information):

$$H(X) = E[I(X)] = \sum_S P(x_i) I(x_i) = \sum_S P(x_i) \log \frac{1}{P(x_i)} = E\left[-\log P(X)\right] \tag{3.175}$$

This entropy $H(X)$ is the amount of information required to specify what kind of symbol has occurred on average. It is also the average uncertainty for the symbol. Suppose that the sample space S has an alphabet size $\|S\| = N$. The entropy $H(X)$ attains the maximum value when the pf has a uniform distribution, i.e.:

$$P(x_i) = P(x_j) = \frac{1}{N} \quad \text{for all } i \text{ and } j \tag{3.176}$$

Equation (3.176) can be interpreted to mean that *uncertainty* reaches its maximum level when no outcome is more probable than any other. It can be proved that the entropy $H(X)$ is nonnegative and becomes zero only if the probability function is a deterministic one, i.e.,

$$H(X) \geq 0 \text{ with equality i.f.f. } P(x_i) = 1 \text{ for some } x_i \in S \tag{3.177}$$

There is another very interesting property for the entropy. If we replace the pf of generating symbol x_i in Eq. (3.175) with any other arbitrary pf, the new value is no smaller than the original entropy. That is,

$$H(X) \le E[-\log Q(X)] = -\sum_S P(x_i)\log Q(x_i) \tag{3.178}$$

Equation (3.178) has a very important meaning. It shows that we are more uncertain about the data if we misestimate the distribution governing the data source. The equality for Eq. (3.178) occurs if and only if $P(x_i) = Q(x_i)\ \ 1 \le i \le N$. Equation (3.178), often referred to as *Jensen's inequality*, is the basis for the proof of EM algorithm in Chapter 4. Similarly, Jensen's inequality can be extended to continuous pdf:

$$-\int f_x(x)\log f_x(x)dx \le -\int g_x(x)\log f_x(x)dx \tag{3.179}$$

with equality occurring if and only if $f_x(x) = g_x(x)\ \ \forall x$.

The proof of Eq. (3.178) follows from the fact $\log(x) \le x-1,\ \forall x$, so the following quantity must have an non-positive value.

$$\sum_S P(x_i)\log\frac{Q(x_i)}{P(x_i)} \le \sum_S P(x_i)\left[\frac{Q(x_i)}{P(x_i)} - 1\right] = 0 \tag{3.180}$$

Based on this property, the negation of the quantity in Eq. (3.180) can be used for the measurement of the distance of two probability distributions. Specifically, the *Kullback-Leibler* (KL) *distance* (*relative entropy*, *discrimination*, or *divergence*) is defined as:

$$KL(P \| Q) = E\left[\log\frac{P(X)}{Q(X)}\right] = \sum_S P(x_i)\log\frac{P(x_i)}{Q(x_i)} \tag{3.181}$$

As discussed in Chapter 11, the branching factor of a grammar or language is an important measure of degree of difficulty of a particular task in spoken language systems. This relates to the size of the word list from which a speech recognizer or a natural language processor needs to disambiguate in a given context. According to the entropy definition above, this branching factor estimate (or average choices for an alphabet) is defined as follows:

$$PP(X) = 2^{H(X)} \tag{3.182}$$

$PP(X)$ is called the *perplexity* of source X, since it describes how confusing the grammar (or language) is. The value of perplexity is equivalent to the size of an imaginary equivalent list, whose words are equally probable. The bigger the perplexity, the higher branching factor. To find out the perplexity of English, Shannon devised an ingenious way [22] to estimate the entropy and perplexity of English words and letters. His method is similar to a guessing game where a human subject guesses sequentially the words of a text hidden from him, using the relative frequencies of her/his guesses as the estimates of the probability distribution underlying the source of the text. Shannon's perplexity estimate of English comes out to be about 2.39 for English letters and 130 for English words. Chapter 11 has a detailed description on the use of perplexity for language modeling.

3.4.2. Conditional Entropy

Now let us consider transmission of symbols through an information channel. Suppose the input alphabet is $X=(x_1,x_2,\ldots,x_s)$, the output alphabet is $Y=(y_1,y_2,\ldots,y_t)$, and the information channel is defined by the channel matrix $M_{ij}=P(y_j\mid x_i)$, where $P(y_j\mid x_i)$ is the conditional probability of receiving output symbol y_j when input symbol x_i is sent. Figure 3.17 shows an example of an information channel.

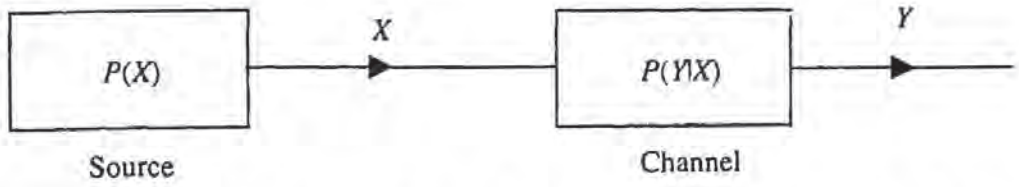


Figure 3.17 Example of information channel. The source is described by source pf $P(X)$ and the channel is characterized by the conditional pf $P(Y|X)$.

Before transmission, the average amount of information, or the uncertainty of the input alphabet X, is the prior entropy $H(X)$.

$$H(X)=\sum_X P(X=x_i)\log\frac{1}{P(X=x_i)} \tag{3.183}$$

where $P(x_i)$ is the prior probability. After transmission, suppose y_j is received; then the average amount of information, or the uncertainty of the input alphabet A, is reduced to the following *posterior* entropy:

$$H(X\mid Y=y_j)=-\sum_X P(X=x_i\mid Y=y_j)\log P(X=x_i\mid Y=y_j) \tag{3.184}$$

where the $P(x_i\mid y_j)$ are the posterior probabilities. Averaging the posterior entropy $H(X\mid y_j)$ over all output symbols y_j leads to the following equation:

$$\begin{aligned}H(X\mid Y)&=\sum_Y P(Y=y_j)H(X\mid Y=y_j)\\&=-\sum_Y P(Y=y_j)\sum_X P(X=x_i\mid Y=y_j)\log P(X=x_i\mid Y=y_j)\\&=-\sum_X\sum_Y P(X=x_i,Y=y_j)\log P(X=x_i\mid Y=y_j)\end{aligned} \tag{3.185}$$

This *conditional entropy*, defined in Eq. (3.185), is the average amount of information or the uncertainty of the input alphabet X given the outcome of the output event Y. Based on the definition of conditional entropy, we derive the following equation:

$$\begin{aligned} H(X,Y) &= -\sum_X \sum_Y P(X=x_i, Y=y_i) \log P(X=x_i, Y=y_i) \\ &= -\sum_X \sum_Y P(X=x_i, Y=y_i) \{\log P(X=x_i) + \log P(Y=y_i \mid X=x_i)\} \\ &= H(X) + H(Y \mid X) \end{aligned} \tag{3.186}$$

Equation (3.186) has an intuitive meaning – the uncertainty about two random variables equals the sum of uncertainty about the first variable and the conditional entropy for the second variable given the first variable is known. Equations (3.185) and (3.186) can be generalized to random vectors **X** and **Y** where each contains several random variables.

It can be proved that the chain rule [Eq. (3.16)] applies to entropy.

$$H(X_1, \cdots, X_n) = H(X_n \mid X_1, \cdots, X_{n-1}) + \cdots + H(X_2 \mid X_1) + H(X_1) \tag{3.187}$$

Finally, the following inequality can also be proved:

$$H(X \mid Y, Z) \le H(X \mid Y) \tag{3.188}$$

with equality i.f.f. X and Z being independent when conditioned on Y. Equation (3.188) basically confirms the intuitive belief that uncertainty decreases when more information is known.

3.4.3. The Source Coding Theorem

Information theory is the foundation for data compressing. In this section we describe *Shannon's source coding theorem*, also known as the *first coding theorem*. In source coding, we are interested in *lossless* compression, which means the compressed information (or symbols) can be recovered (decoded) perfectly. The entropy serves as the upper bound for a source lossless compression.

Consider an information source with alphabet $S = \{0,1,\ldots,N-1\}$. The goal of data compression is to *encode* the output symbols into a string of binary symbols. An interesting question arises: *What is the minimum number of bits required, on the average, to encode the output symbols of the information source?*

Let's assume we have a source that can emit four symbols {0,1,2,3} with equal probability $P(0) = P(1) = P(2) = P(3) = 1/4$. Its entropy is 2 bits as illustrated in Eq. (3.189):

$$H(S) = \sum_{i=0}^{3} P(i) \log_2 \frac{1}{P(i)} = 2 \tag{3.189}$$

It is obvious that 2 bits per symbol is good enough to encode this source. A possible binary code for this source is {00, 01, 10, 11}. It could happen, though some symbols are more likely than others, for example, $P(0) = 1/2$, $P(1) = 1/4$, $P(2) = 1/8$, $P(3) = 1/8$. In this case the entropy is only 1.75 bits. One obvious idea is to use fewer bits for lower values that are frequently used and more bits for larger values that are rarely used. To represent this

source we can use a variable-length code {0,10,110,111}, where no codeword is a prefix for the rest and thus a string of 0s and 1s can be uniquely broken into those symbols. The encoding scheme with such a property is called *uniquely decipherable* (or instantaneous) coding, because as soon as the decoder observes a sequence of codes, it can decisively determine the sequence of the original symbols. If we let $r(x)$ be the number of bits (length) used to encode symbol x, the average rate R of bits per symbol used for encoding the information source is:

$$R = \sum_x r(x)P(x) \tag{3.190}$$

In our case, R is 1.75 bits as shown in Eq. (3.191):

$$R = 0.5\times1 + 0.25\times2 + 0.125\times3 + 0.125\times3 = 1.75 \tag{3.191}$$

Such variable-length coding strategy is called *Huffman coding*. Huffman coding belongs to *entropy coding* because it matches the entropy of the source. In general, *Shannon's source coding* theorem says that a source cannot be coded with fewer bits than its entropy. We will skip the proof here. Interested readers can refer to [3, 15, 17] for the detailed proof. This theorem is consistent with our intuition because the entropy measure is exactly the information content of the information measured in bits. If the entropy increases, then uncertainty increases, resulting in a large amount of information. Therefore, it takes more bits to encode the symbols. In the case above, we are able to match this rate, but, in general, this is impossible, though we can get arbitrarily close to it. The Huffman code for this source offers a compression rate of 12.5% relative to the code designed for the uniform distribution.

Shannon's source coding theorem establishes not only the lower bound for lossless compression but also the upper bound. Let $\lceil x \rceil$ denote the smallest integer that greater or equal to x. As in the similar procedure above, we can make the code length assigned to source output x equal to

$$l(x) = \lceil -\log P(x) \rceil \tag{3.192}$$

The average length L satisfies the following inequality:

$$L = \sum_x l(x)P(x) < \sum_x [1 - \log P(x)]P(x) = 1 + H(X) \tag{3.193}$$

Equation (3.193) means that the average rate R only exceeds the value of entropy by less than one bit.

L can be made arbitrarily close to the entropy by block coding. Instead of encoding single output symbols of the information source, one can encode each block of length n. Let's assume the source is memoryless, so $X_1, X_2, \ldots, X_n$ are independent. According to Eq. (3.193), the average rate R for this block code satisfies:

$$L < 1 + H(X_1, X_2, \ldots, X_n) = 1 + nH(X) \tag{3.194}$$

This makes the average number of bits per output symbol, L/n, satisfy

$$\lim_{n\to\infty}\frac{1}{n}L \le H(X) \tag{3.195}$$

In general, Huffman coding arranges the symbols in order of decreasing probability, assigns the bit 0 to the symbol of highest probability and the bit 1 to what is left, and proceeds the same way for the second highest probability value (which now has a code 10) and iterate. This results in 2.25 bits for the uniform distribution case, which is higher than the 2 bits we obtain with equal-length codes.

Lempel-Ziv coding is a coding strategy that uses correlation to encode *strings* of symbols that occur frequently. Although it can be proved to converge to the entropy, its convergence rate is much slower [27]. Unlike Huffman coding, Lempel-Ziv coding is independent of the distribution of the source; i.e., it needs not be aware of the distribution of the source before encoding. This type of coding scheme is often referred to as *universal* encoding scheme.

3.4.4. Mutual Information and Channel Coding

Let's review the information channel illustrated in Figure 3.17. An intuitively plausible measure of the average amount of information provided by the random event Y about the random event X is the average difference between the number of bits it takes to specify the outcome of X when the outcome of Y is not known and the outcome of Y is known. *Mutual information* is defined as the difference in the entropy of X and the conditional entropy of X given Y:

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= \sum_X P(x_i)\log\frac{1}{P(x_i)} - \sum_X\sum_Y P(x_i,y_j)\log\frac{1}{P(x_i \mid y_j)} \\ &= \sum_X\sum_Y P(x_i,y_j)\log\frac{P(x_i \mid y_j)}{P(x_i)} = \sum_X\sum_Y P(x_i,y_j)\log\frac{P(x_i,y_j)}{P(x_i)P(y_i)} \\ &= E\left[\log\frac{P(X,Y)}{P(X)P(Y)}\right] \end{aligned} \tag{3.196}$$

$I(X;Y)$ is referred to as the mutual information between X and Y. $I(X;Y)$ is symmetrical; i.e., $I(X;Y) = I(Y;X)$. The quantity $P(x,y)/P(x)P(y)$ is often referred to as the *mutual information between symbol x and y*. $I(X;Y)$ is bounded:

$$0 \le I(X;Y) \le \min\left[H(X), H(Y)\right] \tag{3.197}$$

$I(X;Y)$ reaches the minimum value (zero) when the random variables X and Y are independent.

Mutual information represents the information obtained (or the reduction in uncertainty) through a channel by observing an output symbol. If the information channel is noiseless, the input symbol can be determined definitely by observing an output symbol. In this case, the conditional entropy $H(X|Y)$ equals zero and it is called a *noiseless channel*. We obtain the maximum mutual information $I(X; Y) = H(X)$. However, the information channel is generally *noisy* so that the conditional entropy $H(X|Y)$ is not zero. Therefore, maximizing the mutual information is equivalent to obtaining a low-noise information channel, which offers a closer relationship between input and output symbols.

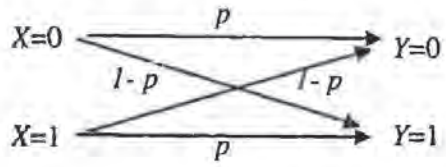


Figure 3.18 A binary channel with two symbols.

Let's assume that we have a binary channel, a channel with a binary input and output as shown in Figure 3.18. Associated with each output are a probability p that the output is correct, and a probability $(1-p)$ that it is not, so that the channel is *symmetric*.

If we observe a symbol $Y = 1$ at the output, we don't know for sure what symbol X was transmitted, though we know $P(X=1|Y=1)=p$ and $P(X=0|Y=1)=(1-p)$, so that we can measure our uncertainty about X by its conditional entropy:

$$H(X|Y=1) = -p\log p-(1-p)\log(1-p) \tag{3.198}$$

If we assume that our source X has a uniform distribution, $H(X|Y)=H(X|Y=1)$ as shown in Eq. (3.198) and $H(X) = 1$. The mutual information between X and Y is given by

$$I(X,Y) = H(X)-H(X|Y)=1+p\log p+(1-p)\log(1-p) \tag{3.199}$$

It measures the information that Y carries by about X. The channel capacity C is the maximum of the mutual information over all distributions of X. That is,

$$C=\max_{P(x)} I(X;Y) \tag{3.200}$$

The channel capacity C can be attained by varying the distribution of the information source until the mutual information is maximized for the channel. The channel capacity C can be regarded as a channel that can transmit at most C bits of information per unit of time. *Shannon's channel coding* theorem says that for a given channel there exists a code that permits error-free transmission across the channel, provided that $R \le C$, where R is the rate of the communication system, which is defined as the number of bits per unit of time being transmitted by the communication system. Shannon's channel coding theorem states the fact that *arbitrarily reliable communication is possible at any rate below channel capacity*.

Figure 3.19 illustrates a transmission channel with the source encoder and destination decoder. The source encoder will encode the source symbol sequence $\mathbf{x} = x_1, x_2, \ldots, x_n$ into

channel input sequence $y_1, y_2, \ldots, y_k$. The destination decoder takes the output sequence $z_1, z_2, \ldots, z_k$ from the channel and converts it into the estimates of the source output $\overline{\mathbf{x}} = \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n$. The goal of this transmission is to make the probability of correct decoding $P(\overline{\mathbf{x}} = \mathbf{x})$ asymptotically close to 1 while keeping the compression ratio $\Re = n/k$ as large as possible. *Shannon's source-channel coding* theorem (also referred to as *Shannon's second coding* theorem) says that it is possible to find an encoder-decoder pair of rate $\Re$ for a noisy information channel, provided that $\Re \times H(X) \leq C$.

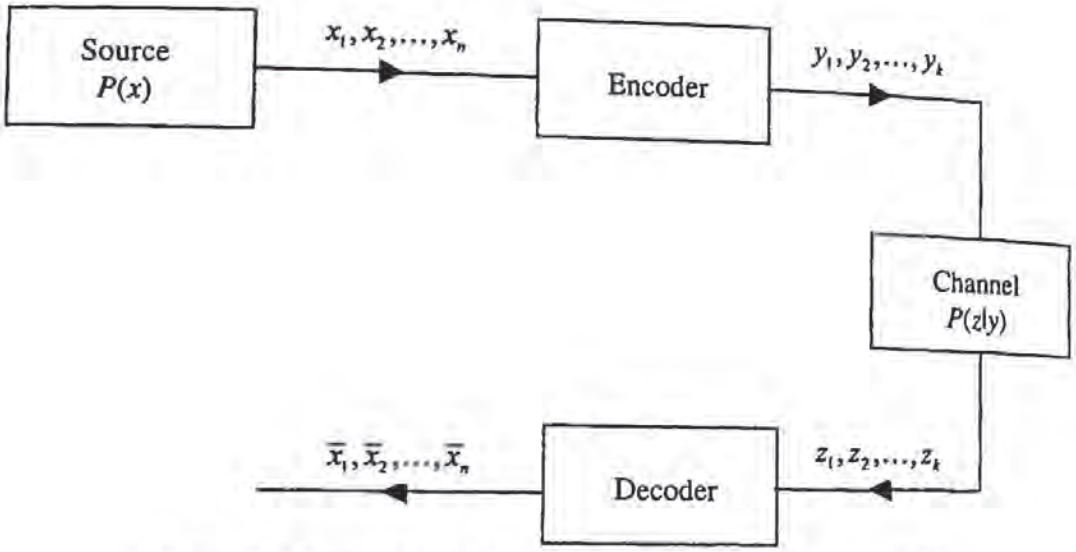


Figure 3.19 Transmission of information through a noisy channel [15].

Because of channel errors, speech coders need to provide error correction codes that will decrease the bit rate allocated to the speech. In practice, there is a tradeoff between the bit rate used for source coding and the bit rate for channel coding. In Chapter 7 we will describe speech coding in great detail.

3.5. HISTORICAL PERSPECTIVE AND FURTHER READING

The idea of uncertainty and probability can be traced all the way back to about 3500 B.C., when games of chance played with bone objects were developed in Egypt. Cubical dice with markings virtually identical to modern dice have been found in Egyptian tombs dating around 2000 B.C. Gambling with dice played an important part in the early development of probability theory. Modern mathematical theory of probability is believed to have been started by the French mathematicians Blaise Pascal (1623-1662) and Pierre Fermat (1601-1665) when they worked on certain gambling problems involving dice. English mathematician Thomas Bayes (1702-1761) was first to use probability inductively and established a mathematical basis for probability inference, leading to what is now known as Bayes' theorem. The theory of probability has developed steadily since then and has been widely ap-

plied in diverse fields of study. There are many good textbooks on probability theory. The book by DeGroot [6] is an excellent textbook for both probability and statistics which covers all the necessary elements for engineering majors. The authors also recommend [14], [19], or [24] for interested readers.

Estimation theory is a basic subject in statistics covered in textbooks. The books by DeGroot [6], Wilks [26] and Hoel [13] offer excellent discussions of estimation theory. They all include comprehensive treatments for maximum likelihood estimation and Bayesian estimation. Maximum likelihood estimation was introduced in 1912 by R. A. Fisher (1890-1962) and has been applied to various domains. It is arguably the most popular parameter estimation method due to its intuitive appeal and excellent performance with large training samples. The EM algorithm in Chapter 4 and the estimation of hidden Markov models in Chapter 8 are based on the principle of MLE. The use of prior distribution in Bayesian estimation is very controversial in statistics. Some statisticians adhere to the Bayesian philosophy of statistics by taking the Bayesian estimation view of the parameter Φ having a probability distribution. Others, however, believe that in many problems Φ is not a random variable but rather a fixed number whose value is unknown. Those statisticians believe that a prior distribution can be assigned to a parameter Φ only when there is extensive prior knowledge of the past; thus the non-informative priors are completely ruled out. Both groups of statisticians agree that whenever a meaningful prior distribution can be obtained, the theory of Bayesian estimation is applicable and useful. The books by DeGroot [6] and Poor [20] are excellent for learning the basics of Bayesian and MAP estimations. Bayesian and MAP adaptation are particularly powerful when the training samples are sparse. Therefore, they are often used for adaptation where the knowledge of prior distribution can help to adapt the model to a new but limited training set. The speaker adaptation work done by Brown et al. [2] first applied Bayesian estimation to speech recognition and [9] is another good paper on using MAP for hidden Markov models. References [4], [16] and [14] have extensive studies of different conjugate prior distributions for various standard distributions. Finally, [1] has an extensive reference for Bayesian estimation.

Significance testing is an essential tool for statisticians to interpret all the statistical experiments. Neyman and Pearson provided some of the most important pioneering work in hypotheses testing [18]. There are many different testing methods presented in most statistics books. The χ^2 test, invented in 1900 by Karl Pearson, is arguably the most widely used testing method. Again, the textbook by DeGroot [6] is an excellent source for the basics of testing and various testing methods. The authors recommend [7] as an interesting book that uses many real-world examples to explain statistical theories and methods, particularly the significance testing.

Information theory first appeared in Claude Shannon's historical paper: *A Mathematical Theory of Communication* [21]. In it, Shannon, analyzed communication as the transmission of a message from a source through a channel to a receiver. In order to solve the problem he created a new branch of applied mathematics – *information and coding theory*. IEEE published a collection of Shannon's papers [23] containing all of his published works, as well as many that have never been published. Those published include his classic papers on information theory and switching theory. Among the unpublished works are his once-

secret wartime reports, his Ph.D. thesis on population genetics, unpublished Bell Labs memoranda, and a paper on the theory of juggling. The textbook by McEliece [17] is excellent for learning all theoretical aspects of information and coding theory. However, it might be out of print now. Instead, the books by Hamming [12] and Cover [3] are two current great references for information and coding theory. Finally, F. Jelinek's *Statistical Methods for Speech Recognition* [15] approaches the speech recognition problem from an information-theoretic aspect. It is a useful book for people interested in both topics.

REFERENCES

[1] Bernardo, J.M. and A.F.M. Smith, *Bayesian Theory*, 1996, New York, John Wiley.

[2] Brown, P., C.-H. Lee, and J. Spohrer, "Bayesian Adaptation in Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1983, Boston, MA, pp. 761-764.

[3] Cover, T.M. and J.A. Thomas, *Elements of Information Theory*, 1991, New York, John Wiley and Sons.

[4] DeGroot, M.H., *Optimal Statistical Decisions*, 1970, New York, NY, McGraw-Hill.

[5] DeGroot, M.H., *Probability and Statistics*, Addison-Wesley Series in Behavioral Science: Quantitive Methods, F. Mosteller, ed., 1975, Reading, MA, Addison-Wesley Publishing Company.

[6] DeGroot, M.H., *Probability and Statistics*, 2nd ed, Addison-Wesley Series in Behavioral Science: Quantitive Methods, F. Mosteller, ed., 1986, Reading, MA, Addison-Wesley Publishing Company.

[7] Freedman, D., *et al.*, *Statistics*, 2nd ed, 1991, New York, W. W. Norton & Company, Inc.

[8] Gales, M.J., *Model Based Techniques for Noise Robust Speech Recognition*, PhD Thesis in Engineering Department 1995, Cambridge University.

[9] Gauvain, J.L. and C.H. Lee, "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(2), pp. 291-298.

[10] Gillett, G.E., *Introduction to Operations Research: A Computer-Oriented Algorithmic Approach*, McGraw-Hill Series in Industrial Engineering and Management Science, J. Riggs, ed., 1976, New York, McGraw-Hill.

[11] Gillick, L. and S.J. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, Glasgow, Scotland, UK, IEEE, pp. 532-535.

[12] Hamming, R.W., *Coding and Information Theory*, 1986, Englewood Cliffs, NJ, Prentice-Hall.

[13] Hoel, P.G., *Introduction to Mathematical Statistics*, 5th ed., 1984, John Wiley & Sons.

[14] Jeffreys, H., *Theory of Probability*, 1961, Oxford University Press.

[15] Jelinek, F., *Statistical Methods for Speech Recognition*, Language, Speech, and Communication, 1998, Cambridge, MA, MIT Press.

[16] Lindley, D.V., "The Use of Prior Probability Distributions in Statistical Inference and Decision," *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1961, Berkeley, CA, Univ. of California Press.

[17] McEliece, R., *The Theory of Information and Coding*, Encyclopedia of Mathematics and Its Applications, eds. R. Gian-Carlo. Vol. 3, 1977, Reading, MA, Addison-Wesley Publishing Company.

[18] Neyman, J. and E.S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses," *Philosophical Trans. of Royal Society*, 1928, **231**, pp. 289-337.

[19] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, 3rd ed., 1991, New York, McGraw-Hill.

[20] Poor, H.V., *An Introduction to Signal Detection and Estimation*, Springer Tests in Electrical Engineering, J.B. Thomas, ed., 1988, New York, Springer-Verlag.

[21] Shannon, C., "A Mathematical Theory of Communication," *Bell System Technical Journal*, 1948, **27**, pp. 379-423, 623-526.

[22] Shannon, C.E., "Prediction and Entropy of Printed English," *Bell System Technical Journal*, 1951, pp. 50-62.

[23] Shannon, C.E., *Claude Elwood Shannon: Collected Papers*, 1993, IEEE.

[24] Viniotis, Y., *Probability and Random Processes for Electrical Engineering*, Outline Series in Electronics & Electrical Engineering, Schaum, ed., 1998, New York, WCB McGraw-Hill.

[25] Wald, A., "Note of Consistency of Maximum Likelihood Estimate," *Ann. Mathematical Statistics*, 1949(20), pp. 595-601.

[26] Wilks, S.S., *Mathematical Statistics*, 1962, New York, John Wiley and Sons.

[27] Ziv, J. and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. on Information Theory*, 1997, **IT-23**, pp. 337-343.

CHAPTER 8

Hidden Markov Models

The hidden Markov model (HMM) is a very powerful statistical method of characterizing the observed data samples of a discrete-time series. Not only can it provide an efficient way to build parsimonious parametric models, but can also incorporate the dynamic programming principle in its core for a unified pattern segmentation and pattern classification of time-varying data sequences. The data samples in the time series can be discretely or continuously distributed; they can be scalars or vectors. The underlying assumption of the HMM is that the data samples can be well characterized as a parametric random process, and the parameters of the stochastic process can be estimated in a precise and well-defined framework. The basic HMM theory was published in a series of classic papers by Baum and his colleagues [4]. The HMM has become one of the most powerful statistical methods for modeling speech signals. Its principles have been successfully used in automatic speech recognition, formant and pitch tracking, speech enhancement, speech synthesis, statistical language modeling, part-of-speech tagging, spoken language understanding, and machine translation [3, 4, 8, 10, 12, 18, 23, 37].

8.1. THE MARKOV CHAIN

A *Markov chain* models a class of random processes that incorporates a minimum amount of memory without being completely memoryless. In this subsection we focus on the discrete-time Markov chain only.

Let $\mathbf{X} = X_1, X_2, \ldots X_n$ be a sequence of random variables from a finite discrete alphabet $O = \{o_1, o_2, \ldots, o_M\}$. Based on the Bayes' rule, we have

$$P(X_1, X_2, \ldots, X_n) = P(X_1)\prod_{i=2}^{n} P(X_i \mid X_1^{i-1}) \tag{8.1}$$

where $X_1^{i-1} = X_1, X_2, \ldots, X_{i-1}$. The random variables X are said to form a first-order Markov chain, provided that

$$P(X_i \mid X_1^{i-1}) = P(X_i \mid X_{i-1}) \tag{8.2}$$

As a consequence, for the first-order Markov chain, Eq. (8.1) becomes

$$P(X_1, X_2, \ldots, X_n) = P(X_1)\prod_{i=2}^{n} P(X_i \mid X_{i-1}) \tag{8.3}$$

Equation (8.2) is also known as the *Markov assumption*. This assumption uses very little memory to model dynamic data sequences: the probability of the random variable at a given time depends only on the value at the preceding time. The Markov chain can be used to model time-invariant (stationary) events if we discard the time index i,

$$P(X_i = s \mid X_{i-1} = s') = P(s \mid s') \tag{8.4}$$

If we associate X_i to a state, the Markov chain can be represented by a finite state process with transition between states specified by the probability function $P(s|s')$. Using this finite state representation, the Markov assumption [Eq. (8.2)] is translated to the following: the probability that the Markov chain will be in a particular state at a given time depends only on the state of the Markov chain at the previous time.

Consider a Markov chain with N distinct states labeled by $\{1, \ldots, N\}$, with the state at time t in the Markov chain denoted as s_t; the parameters of a Markov chain can be described as follows:

$$a_{ij} = P(s_t = j \mid s_{t-1} = i) \quad 1 \le i, j \le N \tag{8.5}$$

$$\pi_i = P(s_1 = i) \quad 1 \le i \le N \tag{8.6}$$

where a_{ij} is the transition probability from state i to state j; and π_i is the initial probability that the Markov chain will start in state i. Both transition and initial probabilities are bound to the constraints:

$$\sum_{j=1}^{N} a_{ij} = 1; \quad 1 \le i \le N$$
$$\sum_{j=1}^{N} \pi_j = 1 \tag{8.7}$$

The Markov chain described above is also called the observable Markov model because the output of the process is the set of states at each time instance t, where each state corresponds to an observable event X_i. In other words, there is one-to-one correspondence between the observable event sequence $\mathbf{X}$ and the Markov chain state sequence $S = s_1, s_2, \ldots s_n$. Consider a simple three-state Markov chain for the Dow Jones Industrial average as shown in Figure 8.1. At the end of each day, the Dow Jones Industrial average may correspond to one of the following states:

state 1 – *up* (in comparison to the index of previous day)
state 2 – *down* (in comparison to the index of previous day)
state 3 – *unchanged* (in comparison to the index of previous day)

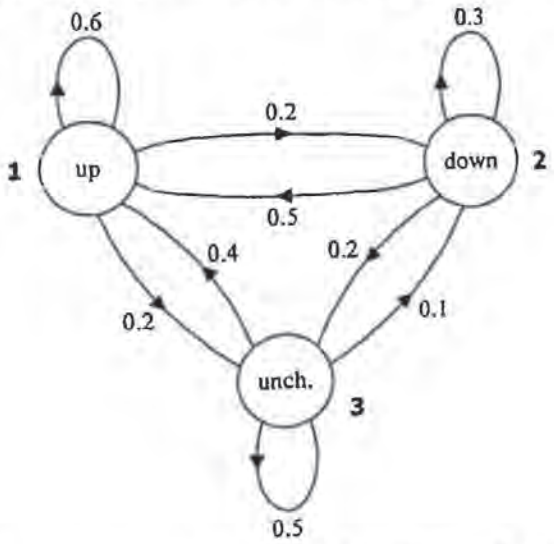


Figure 8.1 A Markov chain for the Dow Jones Industrial average. Three states represent *up*, *down*, and *unchanged*, respectively.

The parameter for this Dow Jones Markov chain may include a state-transition probability matrix

$$A = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

and an initial state probability matrix

$$\pi = (\pi_i)' = \begin{pmatrix} 0.5 \\ 0.2 \\ 0.3 \end{pmatrix}$$

Suppose you would like to find out the probability for five consecutive *up* days. Since the observed sequence of *up-up-up-up-up* corresponds to the state sequence of (1, 1, 1, 1, 1), the probability will be

$$P(5 \text{ consecutive } up \text{ days}) = P(1,1,1,1,1)$$
$$= \pi_1 a_{11} a_{11} a_{11} a_{11} = 0.5 \times (0.6)^4 = 0.0648$$

8.2. DEFINITION OF THE HIDDEN MARKOV MODEL

In the Markov chain, each state corresponds to a deterministically observable event, i.e., the output of such sources in any given state is not random. A natural extension to the Markov chain introduces a non-deterministic process that generates output observation symbols in any given state. Thus, the observation is a probabilistic function of the state. This new model is known as a *hidden Markov model*, which can be viewed as a double-embedded stochastic process with an underlying stochastic process (the state sequence) not directly observable. This underlying process can only be probabilistically associated with another observable stochastic process producing the sequence of features we can observe.

A hidden Markov model is basically a Markov chain where the output observation is a random variable X generated according to a output probabilistic function associated with each state. Figure 8.2 shows a revised hidden Markov model for the Dow Jones Industrial average. You see that each state now can generate all three output observations: *up*, *down*, and *unchanged*, according to its output pdf. This means that there is no longer a one-to-one correspondence between the observation sequence and the state sequence, so you cannot unanimously determine the state sequence for a given observation sequence, i.e., the state sequence is not observable and therefore hidden. This is why the world *hidden* is placed in front of *Markov models*. Although the state of an HMM is hidden, it often contains salient information about the data we are modeling. For example, the first state in Figure 8.2 indicates a *bull* market, and the second state indicates a *bear* market as specified by the output probability in each state. Formally speaking, a hidden Markov model is defined by:

- $O = \{o_1, o_2, \ldots, o_M\}$—An output observation alphabet.[1] The observation symbols correspond to the physical output of the system being modeled. In the case of the Dow Jones Industrial average HMM, the output observation alphabet is the collection of three categories—$O = \{up, down, unchanged\}$.

[1] Although we use the discrete output observation here, we can extend it to the continuous case with a continuous pdf. You can also use vector quantization to map a continuous vector variable into a discrete alphabet set.

- $\Omega = \{1,2,\ldots,N\}$—A set of states representing the state space. Here s_t is denoted as the state at time t. In the case of the Dow Jones Industrial average HMM, the state may indicate a bull market, a bear market, and a stable market.
- $\mathbf{A} = \{a_{ij}\}$—A transition probability matrix, where a_{ij} is the probability of taking a transition from state i to state j, i.e.,

$$a_{ij} = P(s_t = j | s_{t-1} = i) \tag{8.8}$$

- $\mathbf{B} = \{b_i(k)\}$—An output probability matrix,[2] where $b_i(k)$ is the probability of emitting symbol o_k when state i is entered. Let $\mathbf{X} = X_1, X_2, \ldots, X_t, \ldots$ be the observed output of the HMM. The state sequence $S = s_1, s_2, \ldots, s_t, \ldots$ is not observed (hidden), and $b_i(k)$ can be rewritten as follows:

$$b_i(k) = P(X_t = o_k | s_t = i) \tag{8.9}$$

- $\pi = \{\pi_i\}$—A initial state distribution where

$$\pi_i = P(s_0 = i) \quad 1 \le i \le N \tag{8.10}$$

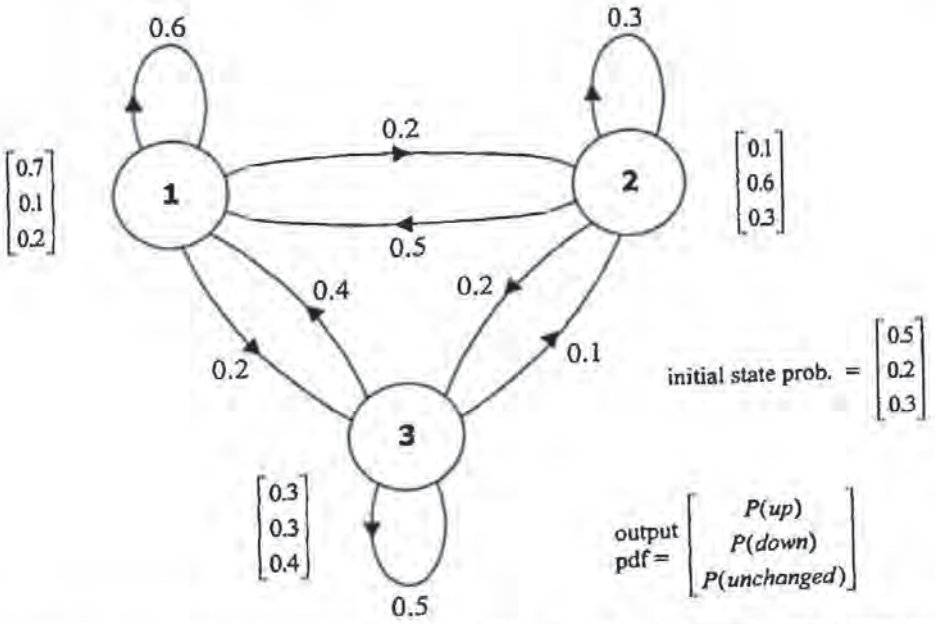


Figure 8.2 A hidden Markov model for the Dow Jones Industrial average. The three states no longer have deterministic meanings as in the Markov chain illustrated in Figure 8.1.

[2] The output distribution can also be transition-dependent. Although these two formulations look different, the state-dependent one can be reformulated as a transition-dependent one with the constraint of all the transitions entering the same state sharing the same output distribution.

Since a_{ij}, $b_i(k)$, and π_i are all probabilities, they must satisfy the following properties:

$$a_{ij} \geq 0, \quad b_i(k) \geq 0, \quad \pi_i \geq 0 \quad \forall \text{ all } i, j, k \tag{8.11}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{8.12}$$

$$\sum_{k=1}^{M} b_i(k) = 1 \tag{8.13}$$

$$\sum_{i=1}^{N} \pi_i = 1 \tag{8.14}$$

To sum up, a complete specification of an HMM includes two constant-size parameters, N and M, representing the total number of states and the size of observation alphabets, observation alphabet O, and three sets (matrices) of probability measures $\mathbf{A}$, $\mathbf{B}$, $\boldsymbol{\pi}$. For convenience, we use the following notation

$$\Phi = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}) \tag{8.15}$$

to indicate the whole parameter set of an HMM and sometimes use the parameter set Φ to represent the HMM interchangeably without ambiguity.

In the first-order hidden Markov model discussed above, there are two assumptions. The first is the *Markov assumption* for the Markov chain.

$$P(s_t | s_1^{t-1}) = P(s_t | s_{t-1}) \tag{8.16}$$

where s_1^{t-1} represents the state sequence $s_1, s_2, \ldots, s_{t-1}$. The second is the *output-independence assumption*:

$$P(X_t | X_1^{t-1}, s_1^t) = P(X_t | s_t) \tag{8.17}$$

where X_1^{t-1} represents the output sequence $X_1, X_2, \ldots, X_{t-1}$. The output-independence assumption states that the probability that a particular symbol is emitted at time t depends only on the state s_t and is conditionally independent of the past observations.

You might argue that these assumptions limit the memory of the first-order hidden Markov models and may lead to model deficiency. However, in practice, they make evaluation, decoding, and learning feasible and efficient without significantly affecting the modeling capability, since those assumptions greatly reduce the number of parameters that need to be estimated.

Given the definition of HMMs above, three basic problems of interest must be addressed before they can be applied to real-world applications.

1. **The Evaluation Problem**—Given a model Φ and a sequence of observations $\mathbf{X} = (X_1, X_2, \ldots, X_T)$, what is the probability $P(\mathbf{X}|\Phi)$, i.e., the probability that the model generates the observations?
2. **The Decoding Problem**—Given a model Φ and a sequence of observations $\mathbf{X} = (X_1, X_2, \ldots, X_T)$, what is the most likely state sequence $\mathbf{S} = (s_0, s_1, s_2, \ldots, s_T)$ in the model that produces the observations?
3. **The Learning Problem**—Given a model Φ and a set of observations, how can we adjust the model parameter $\hat{\Phi}$ to maximize the joint probability (likelihood) $\prod_{\mathbf{X}} P(\mathbf{X}|\Phi)$?

If we could solve the *evaluation* problem, we would have a way of evaluating how well a given HMM matches a given observation sequence. Therefore, we could use HMM to do pattern recognition, since the likelihood $P(\mathbf{X}|\Phi)$ can be used to compute posterior probability $P(\Phi|\mathbf{X})$, and the HMM with highest posterior probability can be determined as the desired pattern for the observation sequence. If we could solve the decoding problem, we could find the best matching state sequence given an observation sequence, or, in other words, we could uncover the hidden state sequence. As discussed in Chapters 12 and 13, these are the basics for the decoding in continuous speech recognition. Last but not least, if we could solve the learning problem, we would have the means to automatically estimate the model parameter Φ from an ensemble of training data. These three problems are tightly linked under the same probabilistic framework. The efficient implementation of these algorithms shares the principle of dynamic programming that we briefly discuss next.

8.2.1. Dynamic Programming and DTW

The dynamic programming concept, also known as *dynamic time warping* (DTW) in speech recognition [40], has been widely used to derive the overall distortion between two speech templates. In these template-based systems, each speech template consists of a sequence of speech vectors. The overall distortion measure is computed from the accumulated distance between two feature vectors that are aligned between two speech templates with minimal overall distortion. The DTW method can warp two speech templates $(\mathbf{x}_1\mathbf{x}_2 \ldots \mathbf{x}_N)$ and $(\mathbf{y}_1\mathbf{y}_2 \ldots \mathbf{y}_M)$ in the time dimension to alleviate nonlinear distortion as illustrated in Figure 8.3.

This is roughly equivalent to the problem of finding the minimum distance in the trellis between these two templates. Associated with every pair (i, j) is a distance $d(i, j)$ between two speech vectors $\mathbf{x}_i$ and $\mathbf{y}_j$. To find the optimal path between starting point (1, 1) and end point (N, M) from left to right, we need to compute the optimal accumulated distance $D(N, M)$. We can enumerate all possible accumulated distance from (1,1) to (N, M) and identify the one that has the minimum distance. Since there are M possible moves for each step from left to right in Figure 8.3, all the possible paths from (1, 1) to (N, M) will be

exponential. Dynamic programming principles can drastically reduce the amount of computation by avoiding the enumeration of sequences that cannot possibly be optimal. Since the same optimal path after each step must be based on the previous step, the minimum distance $D(i, j)$ must satisfy the following equation:

$$D(i,j) = \min_{k}\left[D(i-1,k) + d(k,j)\right] \tag{8.18}$$

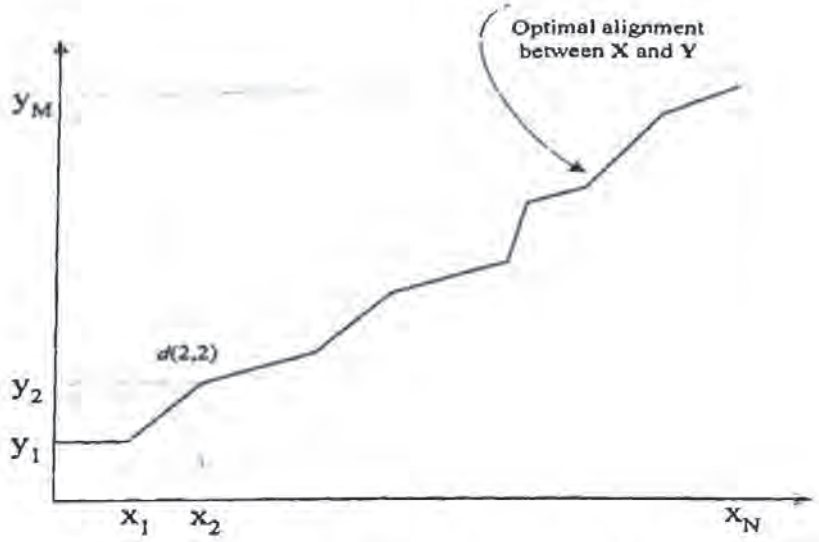


Figure 8.3 Direct comparison between two speech templates $\mathbf{X}=\mathbf{x}_1\mathbf{x}_2...\mathbf{x}_N$ and $\mathbf{Y}=\mathbf{y}_1\mathbf{y}_2...\mathbf{y}_M$.

Equation (8.18) indicates you only need to consider and keep only the best move for each pair although there are M possible moves. The recursion allows the optimal path search to be conducted incrementally from left to right. In essence, dynamic programming delegates the solution recursively to its own sub-problem. The computation proceeds from the small sub-problem ($D(i-1,k)$) to the larger sub-problem ($D(i, j)$). We can identify the optimal match $\mathbf{y}_j$ with respect to $\mathbf{x}_i$ and save the index in a back pointer table $B(i, j)$ as we move forward. The optimal path can be traced back after the optimal path is identified. The algorithm is described in Algorithm 8.1.

The advantage of the dynamic programming lies in the fact that once a sub-problem is solved, the partial result can be stored and never needs to be recalculated. This is a very important principle that you see again and again in building practical spoken language systems.

Speech recognition based on DTW is simple to implement and fairly effective for small-vocabulary speech recognition. Dynamic programming can temporally align patterns to account for differences in speaking rates across talkers as well as across repetitions of the word by the same talker. However, it does not have a principled way to derive an averaged template for each pattern from a large amount of training samples. A multiplicity of reference training tokens is typically required to characterize the variation among different utterances. As such, the HMM is a much better alternative for spoken language processing.

ALGORITHM 8.1: *THE DYNAMIC PROGRAMMING ALGORITHM*

Step 1: Initialization

$D(1,1) = d(1,1), B(1,1) = 1$, for $j = 2,\ldots,M$ compute $D(1,j) = \infty$

Step 2: Iteration

for $i = 2,\ldots,N$ {

for $j = 1,\ldots,M$ compute {

$$D(i,j) = \min_{1 \le p \le M} [D(i-1,p) + d(p,j)]$$

$$B(i,j) = \arg\min_{1 \le p \le M} [D(i-1,p) + d(p,j)] \; \}\}$$

Step 3: Backtracking and Termination

The optimal (minimum) distance is $D(N,M)$ and the optimal path is $(s_1, s_2, \ldots, s_N)$ where $s_N = M$ and $s_i = B(i+1, s_{i+1})$, $i = N-1, N-2, \ldots, 1$

8.2.2. How to Evaluate an HMM—The Forward Algorithm

To calculate the probability (likelihood) $P(\mathbf{X}|\Phi)$ of the observation sequence $\mathbf{X} = (X_1, X_2, \ldots, X_T)$, given the HMM Φ, the most intuitive way is to sum up the probabilities of all possible state sequences:

$$P(\mathbf{X}|\Phi) = \sum_{\text{all } \mathbf{S}} P(\mathbf{S}|\Phi) P(\mathbf{X}|\mathbf{S},\Phi) \tag{8.19}$$

In other words, to compute $P(\mathbf{X}|\Phi)$, we first enumerate all possible state sequences $\mathbf{S}$ of length T, that generate observation sequence $\mathbf{X}$, and then sum all the probabilities. The probability of each path $\mathbf{S}$ is the product of the state sequence probability (first factor) and the joint output probability (the second factor) along the path.

For one particular state sequence $\mathbf{S} = (s_1, s_2, \ldots, s_T)$, where s_1 is the initial state, the state-sequence probability in Eq. (8.19) can be rewritten by applying Markov assumption:

$$P(\mathbf{S}|\Phi) = P(s_1|\Phi) \prod_{t=2}^{T} P(s_t|s_{t-1},\Phi) = \pi_{s_1} a_{s_1 s_2} \ldots a_{s_{T-1} s_T} = a_{s_0 s_1} a_{s_1 s_2} \ldots a_{s_{T-1} s_T} \tag{8.20}$$

where $a_{s_0 s_1}$ denotes π_{s_1} for simplicity. For the same state sequence $\mathbf{S}$, the joint output probability along the path can be rewritten by applying the output-independent assumption:

$$\begin{aligned} P(\mathbf{X}|\mathbf{S},\Phi) &= P(X_1^T|S_1^T,\Phi) = \prod_{t=1}^{T} P(X_t|s_t,\Phi) \\ &= b_{s_1}(X_1) b_{s_2}(X_2) \ldots b_{s_T}(X_T) \end{aligned} \tag{8.21}$$

Substituting Eqs. (8.20) and (8.21) into (8.19), we get:

$$P(\mathbf{X}|\Phi) = \sum_{\text{all S}} P(\mathbf{S}|\Phi)P(\mathbf{X}|\mathbf{S},\Phi)$$
$$= \sum_{\text{all S}} a_{s_0 s_1} b_{s_1}(X_1) a_{s_1 s_2} b_{s_2}(X_2) \ldots a_{s_{T-1} s_T} b_{s_T}(X_T) \tag{8.22}$$

Equation (8.22) can be interpreted as follows. First we enumerate all possible state sequence with length T. For any given state sequence, we start from initial state s_1 with probability π_{s_1} or $a_{s_0 s_1}$. We take a transition from s_{t-1} to s_t with probability $a_{s_{t-1} s_t}$ and generate the observation X_t with probability $b_{s_t}(X_t)$ until we reach the last transition.

However, direct evaluation of Eq. (8.22) according to the interpretation above requires enumeration of $O(N^T)$ possible state sequences, which results in exponential computational complexity. Fortunately, a more efficient algorithm can be used to calculate Eq. (8.22). The trick is to store intermediate results and use them for subsequent state-sequence calculations to save computation. This algorithm is known as the *forward algorithm*.

Based on the HMM assumptions, the calculation of $P(s_t | s_{t-1}, \Phi)P(X_t | s_t, \Phi)$ involves only s_{t-1}, s_t, and X_t, so, it is possible to compute the likelihood with $P(\mathbf{X}|\Phi)$ with recursion on t. Let's define forward probability:

$$\alpha_t(i) = P(X_1^t, s_t = i|\Phi) \tag{8.23}$$

$\alpha_t(i)$ is the probability that the HMM is in state i at time t having generated partial observation X_1^t (namely $X_1 X_2 \ldots X_t$). $\alpha_t(i)$ can be calculated inductively as illustrated in Algorithm 8.2. This inductive procedure shown in Eq. (8.24) can be illustrated in a *trellis*. Figure 8.4 illustrates the computation of forward probabilities α via a trellis framework for the Dow Jones Industrial average HMM shown in Figure 8.2. Given two consecutive *up* days for the Dow Jones Industrial average, we can find the forward probability α based on the model of Figure 8.2. An arrow in Figure 8.4 indicates a transition from its origin state to its destination state. The number inside each cell indicates the forward probability α. We start the α cells from $t = 0$, where the α cells contains exactly the initial probabilities. The other cells are computed in a *time-synchronous* fashion from left to right, where each cell for time t is completely computed before proceeding to time $t+1$. When the states in the last column have been computed, the sum of all probabilities in the final column is the probability of generating the observation sequence. For most speech problems, we need to have the HMM end in some particular exit state (a.k.a final state, S_F), and we thus have $P(\mathbf{X}|\Phi) = \alpha_T(s_F)$.

It is easy to show that the complexity for the forward algorithm is $O(N^2 T)$ rather than exponential. This is because we can make full use of partially computed probabilities for the improved efficiency.

ALGORITHM 8.2: *THE FORWARD ALGORITHM*

Step 1: Initialization

$$\alpha_1(i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$$

Step 2: Induction

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(X_t) \qquad 2 \le t \le T;\ 1 \le j \le N \tag{8.24}$$

Step 3: Termination

$$P(\mathbf{X}|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$$

If it is required to end in the final state, $P(\mathbf{X}|\Phi) = \alpha_T(s_F)$

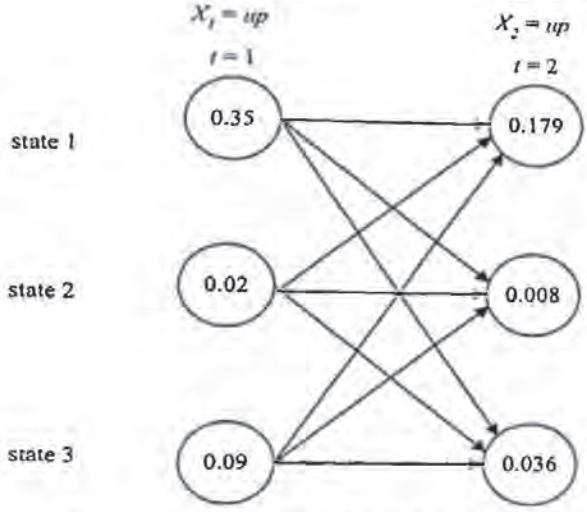


Figure 8.4 The forward trellis computation for the HMM of the Dow Jones Industrial average.

8.2.3. How to Decode an HMM—The Viterbi Algorithm

The forward algorithm, in the previous section, computes the probability that an HMM generates an observation sequence by summing up the probabilities of all possible paths, so it does not provide the best path (or state sequence). In many applications, it is desirable to find such a path. As a matter of fact, finding the best path (state sequence) is the cornerstone for searching in continuous speech recognition. Since the state sequence is hidden (unobserved) in the HMM framework, the most widely used criterion is to find the state sequence that has the highest probability of being taken while generating the observation sequence. In other words, we are looking for the state sequence $\mathbf{S} = (s_1, s_2, \ldots, s_T)$ that maximizes $P(\mathbf{S}, \mathbf{X}|\Phi)$. This problem is very similar to the optimal-path problem in dynamic programming. As a consequence, a formal technique based on dynamic programming, known

as *Viterbi* algorithm [43], can be used to find the best state sequence for an HMM. In practice, the same method can be used to evaluate HMMs that offers an approximate solution close to the case obtained using the forward algorithm described in Section 8.2.2.

The Viterbi algorithm can be regarded as the dynamic programming algorithm applied to the HMM or as a modified forward algorithm. Instead of summing up probabilities from different paths coming to the same destination state, the Viterbi algorithm picks and remembers the best path. To define the best-path probability:

$$V_t(i) = P(X_1^t, S_1^{t-1}, s_t = i \mid \Phi)$$

$V_t(i)$ is the probability of the most likely state sequence at time t, which has generated the observation X_1^t (until time t) and ends in state i. A similar induction procedure for the Viterbi algorithm can be described in Algorithm 8.3.

ALGORITHM 8.3: *THE VITERBI ALGORITHM*

Step 1: Initialization

$V_1(i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$

$B_1(i) = 0$

Step 2: Induction

$$V_t(j) = \underset{1 \le i \le N}{Max}\left[V_{t-1}(i)a_{ij}\right] b_j(X_t) \quad 2 \le t \le T; \quad 1 \le j \le N \tag{8.25}$$

$$B_t(j) = \underset{1 \le i \le N}{Arg\max}\left[V_{t-1}(i)a_{ij}\right] \quad 2 \le t \le T; \quad 1 \le j \le N \tag{8.26}$$

Step 3: Termination

The best score $= \underset{1 \le i \le N}{Max}\left[V_t(i)\right]$

$s_T^* = \underset{1 \le i \le N}{Arg\max}\left[B_T(i)\right]$

Step 4: Backtracking

$s_t^* = B_{t+1}(s_{t+1}^*) \quad t = T-1, T-2, \ldots, 1$

$S^* = (s_1^*, s_2^*, \ldots, s_T^*)$ is the best sequence

This Viterbi algorithm can also be illustrated in a trellis framework similar to the one for the forward algorithm shown in Figure 8.4. Instead of summing up all the paths, Figure 8.5 illustrates the computation of V_t by picking the best path in each cell. The number inside each cell indicates the best score V_t and the best path leading to each cell is indicated by solid lines while the rest of the paths are indicated by dashed line. Again, the computation is done in a *time-synchronous* fashion from left to right. The complexity for the Viterbi algorithm is also $O(N^2T)$.

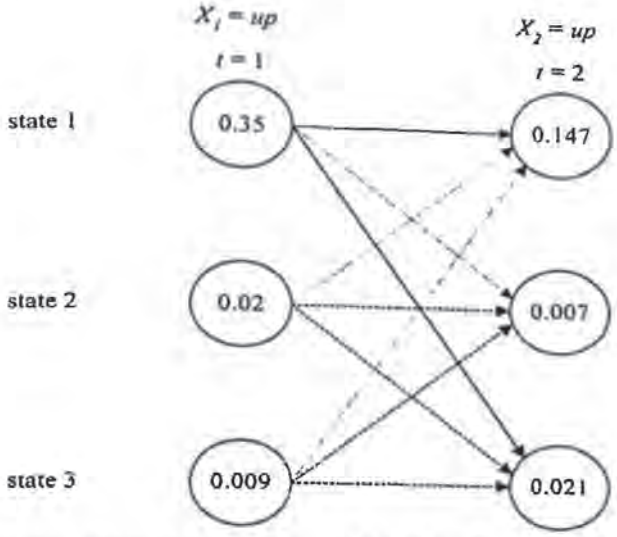


Figure 8.5 The Viterbi trellis computation for the HMM of the Dow Jones Industrial average.

8.2.4. How to Estimate HMM Parameters—Baum-Welch Algorithm

It is very important to estimate the model parameters $\Phi = (\mathbf{A}, \mathbf{B}, \pi)$ to accurately describe the observation sequences. This is by far the most difficult of the three problems, because there is no known analytical method that maximizes the joint probability of the training data in a closed form. Instead, the problem can be solved by the iterative *Baum-Welch* algorithm, also known as the *forward-backward* algorithm.

The HMM learning problem is a typical case of unsupervised learning discussed in Chapter 4, where the data is incomplete because of the hidden state sequence. The EM algorithm is perfectly suitable for this problem. As a matter of fact, Baum and colleagues used the same principle as that of the EM algorithm. Before we describe the formal Baum-Welch algorithm, we first define a few useful terms. In a manner similar to the forward probability, we define backward probability as:

$$\beta_t(i) = P(X_{t+1}^T | s_t = i, \Phi) \tag{8.27}$$

where $\beta_t(i)$ is the probability of generating partial observation X_{t+1}^T (from t+1 to the end) given that the HMM is in state i at time t, $\beta_t(i)$ can then be calculated inductively;

Initialization:

$$\beta_T(i) = 1/N \qquad 1 \le i \le N$$

Induction:

$$\beta_t(i)=\left[\sum_{j=1}^{N} a_{ij} b_j(X_{t+1})\beta_{t+1}(j)\right] \quad t=T-1\ldots1; \quad 1\le i\le N \tag{8.28}$$

The relationship of adjacent α and β (α_{t-1} & α_t and β_t & β_{t+1}) can be best illustrated as shown in Figure 8.6. α is computed recursively from left to right, and β recursively from right to left.

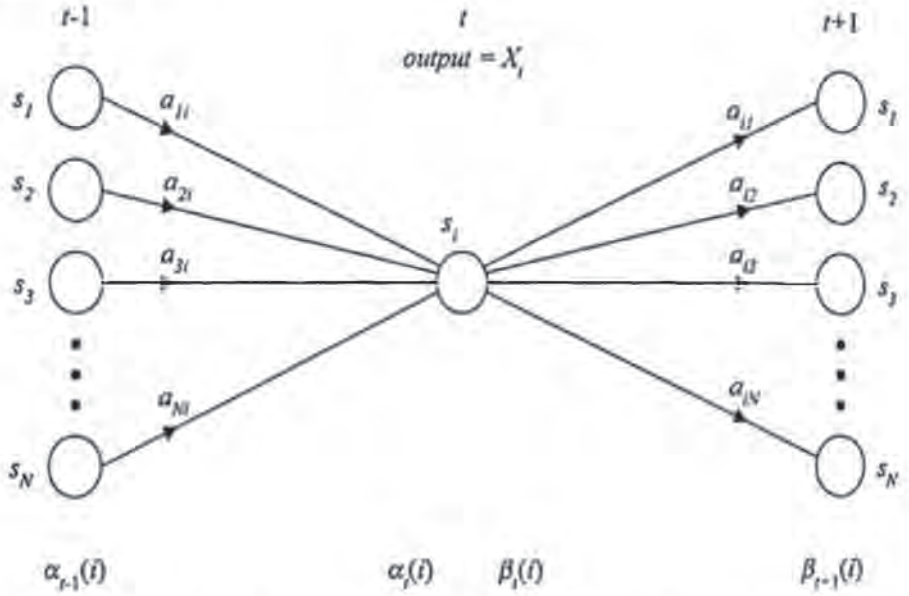


Figure 8.6 The relationship of α_{t-1} and α_t and β_t and β_{t+1} in the forward-backward algorithm.

Next, we define $\gamma_t(i,j)$, which is the probability of taking the transition from state i to state j at time t, given the model and observation sequence, i.e.,

$$\begin{aligned}\gamma_t(i,j) &= P(s_{t-1}=i, s_t=j \mid X_1^T, \Phi)\\ &= \frac{P(s_{t-1}=i, s_t=j, X_1^T \mid \Phi)}{P(X_1^T \mid \Phi)}\\ &= \frac{\alpha_{t-1}(i)a_{ij}b_j(X_t)\beta_t(j)}{\sum_{k=1}^{N}\alpha_T(k)}\end{aligned} \tag{8.29}$$

The equation above can be best illustrated as shown in Figure 8.7.

We can iteratively refine the HMM parameter vector $\Phi=(\mathbf{A},\mathbf{B},\pi)$ by maximizing the likelihood $P(\mathbf{X}|\Phi)$ for each iteration. We use $\hat{\Phi}$ to denote the new parameter vector derived from the parameter vector Φ in the previous iteration. According to the EM algorithm

of Chapter 4, the maximization process is equivalent to maximizing the following Q-function:

$$Q(\Phi,\hat{\Phi}) = \sum_{\text{all } \mathbf{S}} \frac{P(\mathbf{X},\mathbf{S}|\Phi)}{P(\mathbf{X}|\Phi)} \log P(\mathbf{X},\mathbf{S}|\hat{\Phi}) \tag{8.30}$$

where $P(\mathbf{X},\mathbf{S}|\Phi)$ and $\log P(\mathbf{X},\mathbf{S}|\hat{\Phi})$ can be expressed as:

$$P(\mathbf{X},\mathbf{S}|\Phi) = \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(X_t) \tag{8.31}$$

$$\log P(\mathbf{X},\mathbf{S}|\Phi) = \sum_{t=1}^{T} \log a_{s_{t-1}s_t} + \sum_{t=1}^{T} \log b_{s_t}(X_t) \tag{8.32}$$

Equation (8.30) can thus be rewritten as

$$Q(\Phi,\hat{\Phi}) = Q_{\mathbf{a}_i}(\Phi,\hat{\mathbf{a}}_i) + Q_{\mathbf{b}_j}(\Phi,\hat{\mathbf{b}}_j) \tag{8.33}$$

where

$$Q_{\mathbf{a}_i}(\Phi,\hat{\mathbf{a}}_i) = \sum_i \sum_j \sum_t \frac{P(\mathbf{X}, s_{t-1}=i, s_t=j|\Phi)}{P(\mathbf{X}|\Phi)} \log \hat{a}_{ij} \tag{8.34}$$

$$Q_{\mathbf{b}_j}(\Phi,\hat{\mathbf{b}}_j) = \sum_j \sum_k \sum_{t\in X_t=o_k} \frac{P(\mathbf{X}, s_t=j|\Phi)}{P(\mathbf{X}|\Phi)} \log \hat{b}_j(k) \tag{8.35}$$

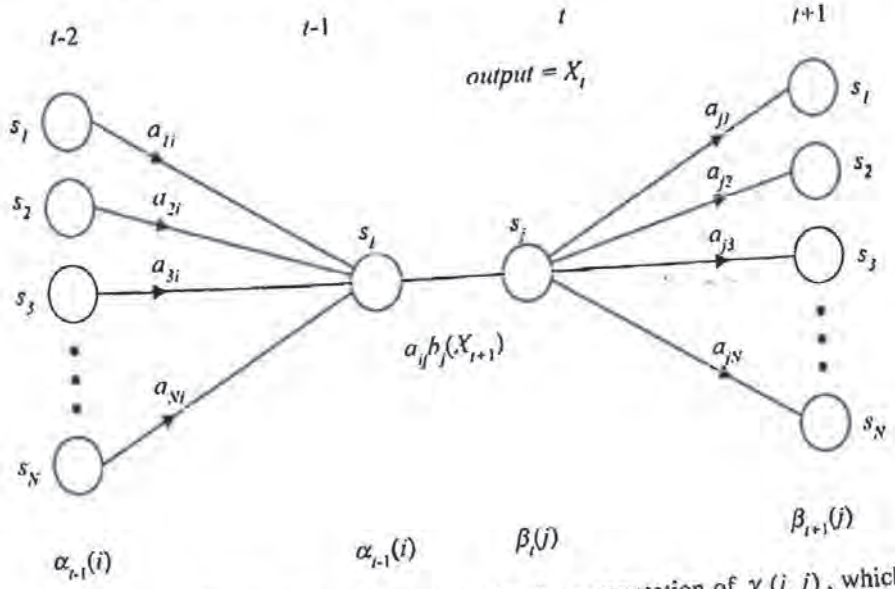


Figure 8.7 Illustration of the operations required for the computation of $\gamma_t(i,j)$, which is the probability of taking the transition from state i to state j at time t.

Since we separate the Q-function into two independent terms, the maximization procedure on $Q(\Phi, \hat{\Phi})$ can be done by maximizing the individual terms separately, subject to probability constraints.

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \forall \text{ all } i \tag{8.36}$$

$$\sum_{k=1}^{M} b_j(k) = 1 \quad \forall \text{ all } j \tag{8.37}$$

Moreover, all these terms in Eqs. (8.34) and (8.35) have the following form:

$$F(x) = \sum_i y_i \log x_i \tag{8.38}$$

where $\sum_i x_i = 1$

By using the Lagrange multipliers, the function above can be proved to achieve maximum value at

$$x_i = \frac{y_i}{\sum_i y_i} \tag{8.39}$$

Using this formation, we obtain the model estimate as[3]:

$$\hat{a}_{ij} = \frac{\frac{1}{P(\mathbf{X}|\Phi)} \sum_{t=1}^{T} P(\mathbf{X}, s_{t-1} = i, s_t = j | \Phi)}{\frac{1}{P(\mathbf{X}|\Phi)} \sum_{t=1}^{T} P(\mathbf{X}, s_{t-1} = i | \Phi)} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_t(i,k)} \tag{8.40}$$

$$\hat{b}_j(k) = \frac{\frac{1}{P(\mathbf{X}|\Phi)} \sum_{t=1}^{T} P(\mathbf{X}, s_t = j | \Phi) \delta(X_t, o_k)}{\frac{1}{P(\mathbf{X}|\Phi)} \sum_{t=1}^{T} P(\mathbf{X}, s_t = j | \Phi)} = \frac{\sum_{t \in X_t = o_k} \sum_i \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_i \gamma_t(i,j)} \tag{8.41}$$

By carefully examining the HMM re-estimation Eqs. (8.40) and (8.41), you can see that Eq. (8.40) is essentially the ratio between the expected number of transitions from state i to state j and the expected number of transitions from state i. For the output probability re-estimation Eq. (8.41), the numerator is the expected number of times the observation data

[3] Notice that the initial probability $\hat{\pi}_i$ can be derived as a special case of the transition probability. $\hat{\pi}_i$ is often fixed (i.e., $\hat{\pi}_i = 1$ for the initial state) for most speech applications.

emitted from state j is the observation symbol o_k, and the denominator is the expected number of times the observation data is emitted from state j.

According to the EM algorithm, the forward-backward (Baum-Welch) algorithm guarantees a monotonic likelihood improvement on each iteration, and eventually the likelihood converges to a local maximum. The forward-backward algorithm can be described in a way similar to the general EM algorithm as shown in Algorithm 8.4.

ALGORITHM 8.4: *THE FORWARD-BACKWARD ALGORITHM*

Step 1: Initialization: Choose an initial estimate Φ.

Step 2: E-step: Compute auxiliary function $Q(\Phi, \hat{\Phi})$ based on Φ.

Step 3: M-step: Compute $\hat{\Phi}$ according to the re-estimation Eqs. (8.40) and (8.41) to maximize the auxiliary Q-function.

Step 4: Iteration: Set $\Phi = \hat{\Phi}$, repeat from step 2 until convergence.

Although the forward-backward algorithm described above is based on one training observation sequence, it can be easily generalized to multiple training observation sequences under the independence assumption between these sequences. To train an HMM from M data sequences is equivalent to finding the HMM parameter vector Φ that maximizes the joint likelihood:

$$\prod_{i=1}^{M} P(\mathbf{X}_i \mid \Phi) \tag{8.42}$$

The training procedure performs the forward-backward algorithm on each independent observation sequence to calculate the expectations (or sometimes referred to as counts) in Eqs. (8.40) and (8.41). These counts in the denominator and numerator, respectively, can be added across M data sequences respectively. Finally, all the model parameters (probabilities) are normalized to make them sum up to one. This constitutes one iteration of Baum-Welch re-estimation; iteration continues until convergence. This procedure is practical and useful because it allows you to train a good HMM in a typical speech recognition scenario where a large amount of training data is available.

For example, if we let $\gamma_t^m(i,j)$ denote the $\gamma_t(i,j)$ from the m^{th} data sequence and T^m denote the corresponding length, Eq. (8.40) can be extended as:

$$\hat{a}_{ij} = \frac{\sum_{m=1}^{M}\sum_{t=1}^{T^m}\gamma_t^m(i,j)}{\sum_{m=1}^{M}\sum_{t=1}^{T^m}\sum_{k=1}^{N}\gamma_t^m(i,k)} \tag{8.43}$$

8.3. CONTINUOUS AND SEMICONTINUOUS HMMS

If the observation does not come from a finite set, but from a continuous space, the discrete output distribution discussed in the previous sections needs to be modified. The difference between the discrete and the continuous HMM lies in a different form of output probability functions. For speech recognition, use of continuous HMMs implies that the quantization procedure to map observation vectors from the continuous space to the discrete space for the discrete HMM is no longer necessary. Thus, the inherent quantization error can be eliminated.

8.3.1. Continuous Mixture Density HMMs

In choosing continuous output probability density functions $b_j(\mathbf{x})$, the first candidate is multivariate Gaussian mixture density functions. This is because they can approximate any continuous probability density function, as discussed in Chapter 3. With M Gaussian mixture density functions, we have:

$$b_j(\mathbf{x}) = \sum_{k=1}^{M} c_{jk} N(\mathbf{x}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) = \sum_{k=1}^{M} c_{jk} b_{jk}(\mathbf{x}) \tag{8.44}$$

where $N(\mathbf{x}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$ or $b_{jk}(\mathbf{x})$ denotes a single Gaussian density function with mean vector $\boldsymbol{\mu}_{jk}$ and covariance matrix $\boldsymbol{\Sigma}_{jk}$ for state j, M denotes the number of mixture-components, and c_{jk} is the weight for the k^{th} mixture component satisfying

$$\sum_{k=1}^{M} c_{jk} = 1 \tag{8.45}$$

To take the same *divide and conquer* approach as Eq. (8.33), we need to express $b_j(\mathbf{x})$ with respect to each single mixture component as:

$$p(\mathbf{X}, \mathbf{S} \mid \Phi) = \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(\mathbf{x}_t) = \sum_{k_1=1}^{M} \sum_{k_2=1}^{M} \cdots \sum_{k_T=1}^{M} \{ \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t k_t}(\mathbf{x}_{s_t}) c_{s_t k_t} \} \tag{8.46}$$

Equation (8.46) can be considered as the summation of densities with all the possible state sequences S and all the possible mixture components K, defined in Ω^T as the T^{th} Cartesian product of $\Omega = \{1, 2, \ldots, M\}$, as follows:

$$p(\mathbf{X}, \mathbf{S}, \mathbf{K} \mid \Phi) = \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t k_t}(\mathbf{x}_t) c_{s_t k_t} \tag{8.47}$$

Therefore, the joint probability density is

$$p(\mathbf{X}|\Phi)=\sum_{\mathbf{S}}\sum_{\mathbf{K}\in\Omega^T} p(\mathbf{X},\mathbf{S},\mathbf{K}|\Phi) \tag{8.48}$$

An auxiliary function $Q(\Phi,\bar{\Phi})$ of two model points, Φ and $\hat{\Phi}$, given an observation $\mathbf{X}$, can be written as:

$$Q(\Phi,\hat{\Phi})=\sum_{\mathbf{S}}\sum_{\mathbf{K}\in\Omega^T}\frac{p(\mathbf{X},\mathbf{S},\mathbf{K}|\Phi)}{p(\mathbf{X}|\Phi)}\log p(\mathbf{X},\mathbf{S},\mathbf{K}|\hat{\Phi}) \tag{8.49}$$

From (8.47), the following decomposition can be shown:

$$\begin{aligned}&\log p(\mathbf{X},\mathbf{S},\mathbf{K}|\hat{\Phi})\\&=\sum_{t=1}^{T}\log\hat{a}_{s_{t-1}s_t}+\sum_{t=1}^{T}\log\hat{b}_{s_t k_t}(\mathbf{x}_t)+\sum_{t=1}^{T}\log\hat{c}_{s_t k_t}\end{aligned} \tag{8.50}$$

Maximization of the likelihood by way of re-estimation can be accomplished on individual parameter sets owing to the separability shown in (8.50). The separation of (8.50) is the key to the increased versatility of a re-estimation algorithm in accommodating mixture observation densities. The auxiliary function can be rewritten in a separated form in a similar manner as Eq. (8.33):

$$\begin{aligned}Q(\Phi,\hat{\Phi})&=\sum_{\mathbf{S}}\sum_{\mathbf{K}}\frac{p(\mathbf{X},\mathbf{S},\mathbf{K}|\Phi)}{p(\mathbf{X}|\Phi)}\log p(\mathbf{X},\mathbf{S},\mathbf{K}|\hat{\Phi})\\&=\sum_{i=1}^{N}Q_{\mathbf{a}_i}(\Phi,\hat{\mathbf{a}}_i)+\sum_{j=1}^{N}\sum_{k=1}^{M}Q_{\mathbf{b}_{jk}}(\Phi,\hat{\mathbf{b}}_{jk})+\sum_{j=1}^{N}\sum_{k=1}^{M}Q_{\mathbf{c}_{jk}}(\Phi,\hat{\mathbf{c}}_{jk})\end{aligned} \tag{8.51}$$

The only difference we have is:

$$Q_{\mathbf{b}_{jk}}(\Phi,\hat{\mathbf{b}}_{jk})=\sum_{t=1}^{T}p(s_t=j,k_t=k|\mathbf{X},\Phi)\log\hat{b}_{jk}(\mathbf{x}_t), \tag{8.52}$$

and

$$Q_{\mathbf{c}_{jk}}(\Phi,\hat{\mathbf{c}}_{jk})=\sum_{t=1}^{T}p(s_t=j,k_t=k|\mathbf{X},\Phi)\log\hat{c}_{jk} \tag{8.53}$$

The optimization procedure is similar to what is discussed in the discrete HMM. The only major difference is $Q_{\mathbf{b}_{jk}}(\Phi,\hat{\mathbf{b}}_{jk})$. Maximization of $Q_{\mathbf{b}_{jk}}(\Phi,\hat{\mathbf{b}}_{jk})$ with respect to $\hat{\mathbf{b}}_{jk}$ is obtained through differentiation with respect to $\{\pi_{jk},\Sigma_{jk}\}$ that satisfies:

$$\nabla_{\hat{\mathbf{b}}_{jk}}Q_{\mathbf{b}_{qk}}(\Phi,\hat{\mathbf{b}}_{jk})=0 \tag{8.54}$$

The solutions are:

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} \frac{p(\mathbf{X}, s_t = j, k_t = k \mid \Phi)}{p(\mathbf{X} \mid \Phi)} \mathbf{x}_t}{\sum_{t=1}^{T} \frac{p(\mathbf{X}, s_t = j, k_t = k \mid \Phi)}{p(\mathbf{X} \mid \Phi)}} = \frac{\sum_{t=1}^{T} \zeta_t(j,k) \mathbf{x}_t}{\sum_{t=1}^{T} \zeta_t(j,k)} \tag{8.55}$$

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{jk} &= \frac{\sum_{t=1}^{T} \frac{p(\mathbf{X}, s_t = j, k_t = k \mid \Phi)}{p(\mathbf{X} \mid \Phi)} (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jk})(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jk})'}{\sum_{t=1}^{T} \frac{p(\mathbf{X}, s_t = j, k_t = k \mid \Phi)}{p(\mathbf{X} \mid \Phi)}} \\ &= \frac{\sum_{t=1}^{T} \zeta_t(j,k) (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jk})(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jk})'}{\sum_{t=1}^{T} \zeta_t(j,k)} \end{aligned} \tag{8.56}$$

where $\zeta_t(j,k)$ is computed as:

$$\zeta_t(j,k) = \frac{p(\mathbf{X}, s_t = j, k_t = k \mid \Phi)}{p(\mathbf{X} \mid \Phi)} = \frac{\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} c_{jk} b_{jk}(\mathbf{x}_t) \beta_t(j)}{\sum_{i=1}^{N} \alpha_T(i)} \tag{8.57}$$

In a similar manner to the discrete HMM, we can derive the reestimation equation for c_{jk} as follows:

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \zeta_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \zeta_t(j,k)} \tag{8.58}$$

8.3.2. Semicontinuous HMMs

Traditionally, the discrete and the continuous mixture density HMMs have been treated separately. In fact, the gap between them can be bridged under some minor assumptions with the so-called semicontinuous or tied-mixture HMM. It assumes the mixture density functions are tied together across all the models to form a set of shared kernels. In the discrete HMM, a VQ codebook is typically used to map the continuous input feature vector $\mathbf{x}$ to o_k, so we can use the discrete output probability distribution $b_j(k)$. The codebook can

be essentially regarded as one of such shared kernels. Accordingly, Eq. (8.44) can be modified as:

$$b_j(\mathbf{x}) = \sum_{k=1}^{M} b_j(k) f(\mathbf{x} \mid o_k) = \sum_{k=1}^{M} b_j(k) N(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{8.59}$$

where o_k is the kth codeword, $b_j(k)$ is the same output probability distribution in the discrete HMM or the mixture weights for the continuous mixture density function, and $N(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ are assumed to be independent of the Markov model and they are shared across all the Markov models with a very large number of mixtures M.

From the discrete HMM point of view, the needed VQ codebook consists of M continuous probability density functions, and each *codeword* has a mean vector and a covariance matrix. Typical quantization produces a codeword index that has minimum distortion to the given continuous observation $\mathbf{x}$. In the semicontinuous HMM, the quantization operation produces values of continuous probability density functions $f(\mathbf{x} \mid o_k)$ for all the codewords o_k. The structure of the semicontinuous model can be roughly the same as that of the discrete one. However, the output probabilities are no longer used directly as in the discrete HMM. In contrast, the VQ codebook density functions, $N(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, are combined with the discrete output probability as in Eq. (8.59). This is a combination of *discrete* model-dependent weighting coefficients with the *continuous* codebook probability density functions. Such a representation can be used to re-estimate the original VQ codebook together with the HMM.

The semicontinuous model also resembles the M-mixture continuous HMM with all the continuous output probability density functions shared among all Markov states. Compared with the continuous mixture HMM, the semicontinuous HMM can maintain the modeling ability of large-mixture probability density functions. In addition, the number of free parameters and the computational complexity can be reduced, because all the probability density functions are tied together, thus providing a good compromise between detailed acoustic modeling and trainability.

In practice, because M is large, Eq. (8.59) can be simplified by using the L most significant values $f(\mathbf{x} \mid o_k)$ for each $\mathbf{x}$ without affecting the performance. Experience has shown that values of L in the range of 1-3% of M are adequate. This can be conveniently obtained during the VQ operations by sorting the VQ output and keeping the L most significant values. Let $\eta(\mathbf{x})$ denote the set of L VQ codewords that has the largest $f(\mathbf{x} \mid o_k)$ for the given $\mathbf{x}$. Then we have:

$$b_j(\mathbf{x}) \cong \sum_{k \in \eta(\mathbf{x})} f(\mathbf{x} \mid o_k) b_j(k) \tag{8.60}$$

Since the number of mixture components in $\eta(\mathbf{x})$ is of lower order than M, Eq. (8.60) can significantly reduce the amount of computation. In fact, $\eta(\mathbf{x})$ is the key to bridge the gap between the continuous and discrete HMM. If $\eta(\mathbf{x})$ contains only the most significant

$f(\mathbf{x} \mid o_k)$ (i.e., only the closest codeword to $\mathbf{x}$), the semicontinuous HMM degenerates to the discrete HMM. On the other hand, a large VQ codebook can be used such that each Markov state could contain a number of its own codewords (a mixture of probability density functions). The discrete output probability $b_{ij}(k)$ thus becomes the mixture weights for each state. This would go to the other extreme, a standard continuous mixture density model. We can also define $\eta(\mathbf{x})$ in such a way that we can have partial tying of $f(\mathbf{x} \mid o_k)$ for different phonetic classes. For example, we can have a phone-dependent codebook.

When we have a tied VQ codebook, re-estimation of these mean vectors and covariance matrices of different models will involve interdependencies. If any observation $\mathbf{x}_t$ (no matter what model it is designated for) has a large value of posterior probability $\zeta_t(j,k)$, it will have a large contribution on re-estimation of parameters of codeword o_k. We can compute the posterior probability for each codeword from $\zeta_t(j,k)$ as defined in Eq. (8.57).

$$\zeta_t(k) = p(\mathbf{x}_t = o_k \mid \mathbf{X}, \Phi) = \sum_j \zeta_t(j,k) \tag{8.61}$$

The re-estimation formulas for the tied mixture can be written as:

$$\hat{\mu}_k = \frac{\sum_{t=1}^{T} \zeta_t(k)\mathbf{x}_t}{\sum_{t=1}^{T} \zeta_t(k)} \tag{8.62}$$

$$\hat{\Sigma}_k = \frac{\sum_{t=1}^{T} \zeta_t(k)(\mathbf{x}_t - \hat{\mu}_k)(\mathbf{x}_t - \hat{\mu}_k)'}{\sum_{t=1}^{T} \zeta_t(k)} \tag{8.63}$$

8.4. PRACTICAL ISSUES IN USING HMMS

While the HMM provides a solid framework for speech modeling, there are a number of issues you need to understand to make effective use of spoken language processing. In this section we point out some of the key issues related to practical applications. For expedience, we mostly use the discrete HMM as our example here.

8.4.1. Initial Estimates

In theory, the re-estimation algorithm of the HMM should reach a local maximum for the likelihood function. A key question is how to choose the right initial estimates of the HMM parameters so that the local maximum becomes the global maximum.

In the discrete HMM, if a probability is initialized to be zero, it will remain zero forever. Thus, it is important to have a reasonable set of initial estimates. Empirical study has shown that, for discrete HMMs, you can use a uniform distribution as the initial estimate. It works reasonably well for most speech applications, though good initial estimates are always helpful to compute the output probabilities.

If continuous mixture density HMMs are used, good initial estimates for the Gaussian density functions are essential. There are a number of ways to obtain such initial estimates:

- You can use the *k*-means clustering procedure, as used in vector quantization clustering. The Markov state segmentation can be derived from the discrete HMM, since it is not very sensitive to the initial parameters. Based on the segmented data, you can use the *k*-means algorithm to derive needed Gaussian mean and covariance parameters. The mixture coefficients can be based on the uniform distribution.
- You can estimate the semicontinuous HMM from the discrete HMM. You simply need to estimate an additional covariance matrix for each VQ codeword and run an additional four or five iterations to refine the semicontinuous HMM based on the discrete HMM, which typically requires four or five iterations from the uniform distribution. When the semi-continuous HMM is trained, you take the top *M codewords*, and in each Markov state use them as the initial Gaussian density functions for the continuous density mixture model.
- You can start training a single mixture Gaussian model. You can compute the parameters from previously segmented data. You can then iteratively split the Gaussian density function in a way similar to VQ codebook generation. You typically need two or three iterations to refine the continuous density after each splitting.

8.4.2. Model Topology

Speech is a time-evolving nonstationary signal. Each HMM state has the ability to capture some quasi-stationary segment in the non-stationary speech signal. A left-to-right topology, as illustrated in Figure 8.8, is a natural candidate to model the speech signal. It has a self-transition to each state that can be used to model contiguous speech features belonging to the same state. When the quasi-stationary speech segment evolves, the left-to-right transition enables a natural progression of such evolution. In such a topology, each state has a state-dependent output probability distribution that can be used to interpret the observable speech signal. This topology is, in fact, one of the most popular HMM structures used in state-of-the-art speech recognition systems. The output probability distribution can be either discrete distributions or a mixture of continuous density functions.

For the left-to-right HMM, the most important parameter in determining the topology is the number of states. The choice of model topology depends on available training data and what the model is used for. If each HMM is used to represent a phone, you need to have at least three to five output distributions. If such a model is used to represent a word, more states are generally required, depending on the pronunciation and duration of the word. For example, the word *tetrahydrocannabino* should have a large number of states in comparison to the word *a*. You may use at least 24 states for the former and three states for the latter. If you have the number of states depending on the duration of the signal, you may want to use 15 to 25 states for each second of speech signal. One exception is that, for silence, you may want to have a simpler topology. This is because silence is stationary, and one or two states will be sufficient.

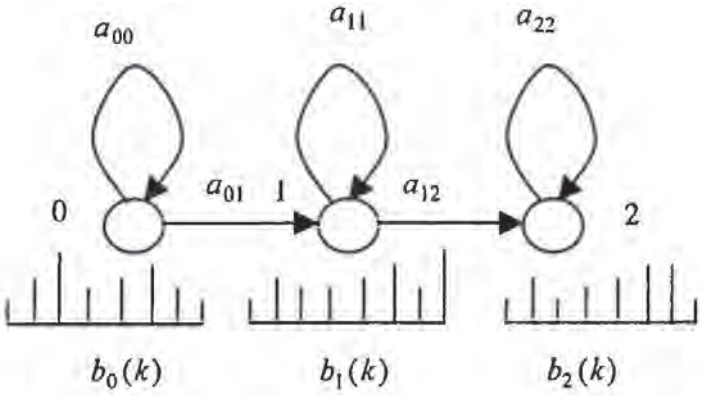


Figure 8.8 A typical hidden Markov model used to model phonemes. There are three states (0-2) and each state has an associated output probability distribution.

In practice, it is convenient to define a *null transition*. This is convenient if we want to simply traverse the HMM without consuming any observation symbol. To incorporate the null arc, you need to slightly modify the basic forward-backward or Viterbi probability equations, provided that no loops of empty transitions exist. If we denote the empty transition between state i and j as a_{ij}^{ε}, they need to satisfy the following constraints:

$$\sum_j a_{ij} + a_{ij}^{\varepsilon} = 1, \forall i \tag{8.64}$$

The forward probability can be augmented as follows:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\mathbf{x}_t) + \sum_{i=1}^{N} \alpha_t(i) a_{ij}^{\varepsilon} \right] \quad 1 \le t \le T; \quad 1 \le j \le N \tag{8.65}$$

Equation (8.65) appears to have a recursion, but it actually uses the value of the same time column of $\alpha_t(i)$, provided that i is already computed, which is easily achievable if we have left-to-right empty transitions without loops of empty transitions.

8.4.3. Training Criteria

The argument for maximum likelihood estimation (MLE) is based on an assumption that the true distribution of speech is a member of the family of distributions used. This amounts to the assertion that the observed speech is genuinely produced by the HMM being used, and the only unknown parameters are the values. However, this can be challenged. Typical HMMs make many inaccurate assumptions about the speech production process, such as the output-independence assumption, the Markov assumption, and the continuous probability density assumption. Such inaccurate assumptions substantially weaken the rationale for maximum likelihood criteria. For instance, although maximum likelihood estimation is consistent (convergence to the true value), it is meaningless to have such a property if the wrong model is used. The true parameters in such cases will be the true parameters of the wrong models. Therefore, an estimation criterion that can work well in spite of these inaccurate assumptions should offer improved recognition accuracy compared with the maximum likelihood criterion. These alternative criteria include the MCE and MMIE, as discussed in Chapter 4. Finally, if we have prior knowledge about the model distribution, we can employ the Bayes' method such as MAP that can effectively combine both the prior and posterior distributions in a consistent way, which is particularly suitable for adaptation or dealing with insufficient training data.

Among all these criteria, MLE remains one of the most widely used, because of its simplicity and superior performance when appropriate assumptions are made about the system design. MCE and MMIE work well for small- to medium-vocabulary speech recognition [2, 26, 36]. You can train a number of other iterations based on the ML estimates. Neither MCE nor MMIE has been found extremely effective for large-vocabulary speech recognition. However, it is possible to combine the MMIE or MCE model with the MLE model for improved performance. This is because the error patterns generated from these different models are not the same. We can decode the test utterance with these different models and vote for the most consistent results [15, 25]. We discuss MAP methods in Chapter 9, since it is mostly helpful for speaker adaptive speech recognition.

8.4.4. Deleted Interpolation

For improved robustness, it is often necessary to combine well trained general models (such as speaker-independent) with those that are less well trained but more detailed (such as speaker-dependent). For example, you can typically improve speech recognition accuracy with speaker-dependent training. Nevertheless, you may not have sufficient data for a particular speaker so it is desirable to use a speaker-independent model that is more general but less accurate to smooth the speaker-dependent model. One effective way to achieve robust-

ness is to interpolate both models with a technique called deleted interpolation, in which the interpolation weights are estimated using cross-validation data. The objective function is to maximize the probability of the model generating the held-out data.

Now, let us assume that we want to interpolate two sets of models, $P_A(\mathbf{x})$ and $P_B(\mathbf{x})$, which can be either discrete probability distributions or continuous density functions, to form an interpolated model $P_{DI}(\mathbf{x})$. The interpolation procedure can be expressed as follows:

$$P_{DI}(\mathbf{x}) = \lambda P_A(\mathbf{x}) + (1-\lambda) P_B(\mathbf{x}) \tag{8.66}$$

where the interpolation weight λ is what we need to derive from the training data.

Consider that we want to interpolate a speaker-independent model $P_A(\mathbf{x})$ with a speaker-dependent model $P_B(\mathbf{x})$. If we use speaker-independent data to estimate the interpolation weight, we may not capture needed speaker-specific information that should be reflected in the interpolation weights. What is worse is that the interpolation weight for the speaker-independent model should be equal to 1.0 if we use the same speaker-independent data from which the model was derived to estimate the interpolation weight. This is because of the MLE criterion. If we use speaker-dependent data instead, we have the weight for the speaker-dependent model equal 1.0 without achieving the desired smoothing effect. Thus the interpolation weights need to be trained using different data or *deleted* data with the so called cross-validation method.

We can have the training data normally divided into M parts, and train a set of $P_A(\mathbf{x})$ and $P_B(\mathbf{x})$ models using the standard EM algorithm from each combination of M-1 parts, with the deleted part serving as the unseen data to estimate the interpolation weights λ. These M sets of interpolation weights are then averaged to obtain the final weights.

ALGORITHM 8.5: *DELETED INTERPOLATION PROCEDURE*

Step 1: Initialize λ with a guessed estimate.

Step 2: Update λ by the following formula:

$$\hat{\lambda} = \frac{1}{M} \sum_{j=1}^{M} \sum_{t=1}^{n_j} \frac{\lambda P_{A-j}(\mathbf{x}_t^j)}{\lambda P_{A-j}(\mathbf{x}_t^j) + (1-\lambda) P_{B-j}(\mathbf{x}_t^j)} \tag{8.67}$$

where $P_{A-j}(\mathbf{x}_t^j)$ and $P_{B-j}(\mathbf{x}_t^j)$ is $P_A(\mathbf{x})$ and $P_B(\mathbf{x})$ estimated by the entire training corpus except part j, the deleted part, respectively; n_j is the total number of data points in part j that have been aligned to estimate the model; and $\mathbf{x}_t^j$ indicates the t-th data point in the j-th set of the aligned data.

Step 3: If the new value $\hat{\lambda}$ is sufficiently close to the previous value λ, stop. Otherwise, go to Step 2.

In fact, the interpolation weights in Eq. (8.66) are similar to the Gaussian mixture weights, although $P_A(\mathbf{x})$ and $P_B(\mathbf{x})$ may not be Gaussian density functions. When we have M sets of data, we can use the same EM algorithm to estimate the interpolation weights as illustrated in Algorithm 8.5.

The deleted interpolation procedure described above can be applied after each training iteration. Then, for the following iteration of training, the learned interpolation weights can be used as illustrated in Eq. (8.66) to compute the forward-backward paths or the Viterbi maximum path. We can also have more than two distributions interpolated together. Deleted interpolation has been widely used in both acoustic and language modeling where smoothing is needed.

8.4.5. Parameter Smoothing

One simple reality for probabilistic modeling is that as many observations as possible are required to reliably estimate model parameters. However, in reality, only a finite amount of training data is available. If the training data are limited, this will result in some parameters being inadequately trained, and classification based on poorly trained models will result in higher recognition error rate. There are many possible solutions to address the problem of insufficient training data:

- You can increase the size of the training data. *There is no data like more data.*
- You can reduce the number of free parameters to be re-estimated. This has its limitations, because a number of significant parameters are always needed to model physical events.
- You can interpolate one set of parameter estimates with another set of parameter estimates, for which an adequate amount of training data exists. Deleted interpolation, discussed in Section 8.4.4, can be used effectively. In the discrete HMM, one simple approach is to set a floor to both the transition probability and the output probability in order to eliminate possible zero estimates. The same principle applies to the SCHMM as well as the mixing coefficients of the continuous density HMM. Parameter flooring can be regarded as a special case of interpolation with the uniform distribution.
- You can tie parameters together to reduce the number of free parameters. The SCHMM is a typical example of such parameter-tying techniques.

For the continuous mixture HMM, you need to pay extra attention to smoothing the covariance matrices. There are a number of techniques you can use:

- You can interpolate the covariance matrix with those that are better trained or a priori via the MAP method.

- You can tie the Gaussian covariance matrices across different mixture components or across different Markov states. A very general shared Gaussian density model is discussed in [20].
- You can use the diagonal covariance matrices if the correlation among feature coefficients is weak, which is the case if you use uncorrelated features such as the MFCC.
- You can combine these methods together.

In practice, we can reduce the speech recognition error rate by 5-20% with various smoothing techniques, depending on the available amount of training data.

8.4.6. Probability Representations

When we compute the forward and backward probabilities in the forward-backward algorithm, they will approach zero in exponential fashion if the observation sequence length, T, becomes large enough. For sufficiently large T, the dynamic range of these probabilities will exceed the precision range of essentially any machine. Thus, in practice, it will result in underflow on the computer if probabilities are represented directly. We can resolve this implementation problem by scaling these probabilities with some scaling coefficient so that they remain within the dynamic range of the computer. All of these scaling coefficients can be removed at the end of the computation without affecting the overall precision.

For example, let $\alpha_t(i)$ be multiplied by a scaling coefficient, S_t:

$$S_t = 1/\sum_i \alpha_t(i) \tag{8.68}$$

so that $\sum_i S_t\alpha_t(i) = 1$ for $1 \le t \le T$. $\beta_t(i)$ can also be multiplied by S_t for $1 \le t \le T$. The recursion involved in computing the forward and backward variables can be scaled at each stage of time t by S_t. Notice that $\alpha_t(i)$ and $\beta_t(i)$ are computed recursively in exponential fashion; therefore, at time t, the total scale factor applied to the forward variable $\alpha_t(i)$ is

$$Scale_\alpha(t) = \prod_{k=1}^{t} S_k \tag{8.69}$$

and the total scale factor applied to the backward variable $\beta_t(i)$ is

$$Scale_\beta(t) = \prod_{k=t}^{T} S_k \tag{8.70}$$

This is because the individual scaling factors are multiplied together in the forward and backward recursion. Let $\alpha'_t(i)$, $\beta'_t(i)$, and $\gamma'_t(i,j)$ denote their corresponding scaled variables, respectively. Note that

$$\sum_i \alpha'_T(i) = Scale_\alpha(T)\sum_i \alpha_T(i) = Scale_\alpha(T)P(\mathbf{X}\,|\,\Phi) \tag{8.71}$$

The scaled intermediate probability, $\gamma'_t(i,j)$, can then be written as:

$$\gamma'_t(i,j) = \frac{Scale_\alpha(t-1)\alpha_{t-1}(i)a_{ij}b_j(X_t)\beta_t(j)Scale_\beta(t)}{Scale_\alpha(T)\sum_{i=1}^{\Lambda}\alpha_T(i)} = \gamma_t(i,j) \tag{8.72}$$

Thus, the intermediate probabilities can be used in the same way as the unscaled probabilities, because the scaling factor is cancelled out in Eq. (8.72). Therefore, re-estimation formulas can be kept exactly except that $P(\mathbf{X}\,|\,\Phi)$ should be computed according to

$$P(\mathbf{X}\,|\,\Phi) = \sum_i \alpha'_T(i) / Scale_\alpha(T) \tag{8.73}$$

In practice, the scaling operation need not be performed at every observation time. It can be used at any scaling interval for which the underflow is likely to occur. In the unscaled interval, $Scale_\alpha$ can be kept as unity.

An alternative way to avoid underflow is to use a logarithmic representation for all the probabilities. This not only ensures that scaling is unnecessary, as underflow cannot happen, but also offers the benefit that integers can be used to represent the logarithmic values, thereby changing floating point operations to fixed point ones, which is particularly suitable for Viterbi-style computation, as Eq. (8.25) requires no probability addition.

In the forward-backward algorithm we need to have probability addition. We can keep a table on $\log_b P_2 - \log_b P_1$. If we represent probability P by $\log_b P$, more precision can be obtained by setting b closer to unity. Let us assume that we want to add P_1 and P_2 and that $P_1 \geq P_2$. We have:

$$\log_b(P_1 + P_2) = \log_b P_1 + \log_b(1 + b^{\log_b P_2 - \log_b P_1}) \tag{8.74}$$

If P_2 is many orders of magnitude smaller than P_1, adding the two numbers will just result in P_1. We could store all possible values of $\log_b(1+b^x)$ in a table. Using logarithms introduces errors for addition operation. In practice, double-precision float representation can be used to minimize the impact of the precision problems.

8.5. HMM LIMITATIONS

There are a number of limitations in the conventional HMMs. For example, HMMs assume the duration follows an exponential distribution, the transition probability depends only on the origin and destination, and all observation frames are dependent only on the state that generated them, not on neighboring observation frames. Researchers have proposed a number of techniques to address these limitations, albeit these solutions have not significantly improved speech recognition accuracy for practical applications.

8.5.1. Duration Modeling

One major weakness of conventional HMMs is that they do not provide an adequate representation of the temporal structure of speech. This is because the probability of state occupancy decreases exponentially with time as shown in Eq. (8.75). The probability of t consecutive observations in state i is the probability of taking the self-loop at state i for t times, which can be written as

$$d_i(t) = a_{ii}^t(1-a_{ii}) \tag{8.75}$$

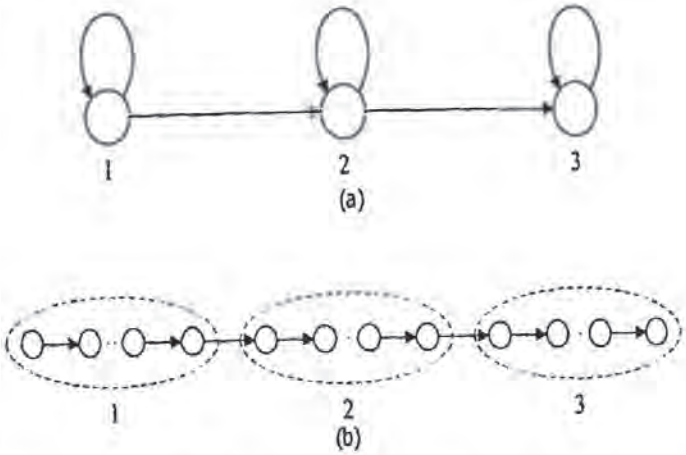


Figure 8.9 A standard HMM (a) and its corresponding explicit duration HMM (b) where the self transitions are replaced with the explicit duration probability distribution for each state.

An improvement to the standard HMM results from the use of HMMs with an explicit time duration distribution for each state [30, 39]. To explain the principle of time duration modeling, a conventional HMM with exponential state duration density and a time duration HMM with specified state duration densities (which can be either a discrete distribution or a continuous density) are illustrated in Figure 8.9. In (a), the state duration probability has an exponential form as in Eq. (8.75). In (b), the self-transition probabilities are replaced with an explicit duration probability distribution. At time t, the process enters state i for duration τ with probability density $d_i(\tau)$, during which the observations $X_{t+1}, X_{t+2} \ldots X_{t+\tau}$ are generated. It then transfers to state j with transition probability a_{ij} only after the appropriate τ observations have occurred in state i. Thus, by setting the time duration probability density to be the exponential density of Eq. (8.75) the time duration HMM can be made equivalent to the standard HMM. The parameters $d_i(\tau)$ can be estimated from observations along with the other parameters of the HMM. For expedience, the duration density is usually truncated at a maximum duration value T_d. To re-estimate the parameters of the HMM with time duration modeling, the forward recursion must be modified as follows:

$$\alpha_t(j) = \sum_{\tau} \sum_{i, i \neq j} \alpha_{t-\tau}(i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t-\tau+l}) \tag{8.76}$$

where the transition from state i to state j depends not only upon the transition probability a_{ij} but also upon all the possible time periods τ that may occur in state j. Intuitively, Eq. (8.76) illustrates that when state j is reached from previous states i, the observations may stay in state j for a period of τ with duration density $d_j(\tau)$, and each observation emits its own output probability. All possible durations must be considered, which leads to summation with respect to τ. The independence assumption of observations results in the Π term of the output probabilities. Similarly, the backward recursion can be written as:

$$\beta_t(i) = \sum_{\tau} \sum_{j, j \neq i} a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t+l}) \beta_{t+\tau}(j) \tag{8.77}$$

The modified Baum-Welch algorithm can then be used based on Eq. (8.76) and (8.77). The proof of the re-estimation algorithm can be based on the modified Q-function except that $P(\mathbf{X}, \mathbf{S}|\Phi)$ should be replaced with $P(\mathbf{X}, \mathbf{S}, \mathbf{T}|\Phi)$, which denotes the joint probability of observation, $\mathbf{X}$, state sequence, $\mathbf{S} = \{s_1, s_2 \ldots, s_k \ldots s_{N_s}\}$ in terms of state s_k with time duration τ_k, and the corresponding duration sequence, $\mathbf{T} = \{\tau_1, \tau_2, \ldots \tau_k \ldots \tau_{N_s}\}$.

$$Q(\Phi, \hat{\Phi}) = \frac{1}{P(\mathbf{X}|\Phi)} \sum_{\mathbf{T}} \sum_{\mathbf{S}} P(\mathbf{X}, \mathbf{S}, \mathbf{T}|\Phi) \log P(\mathbf{X}, \mathbf{S}, \mathbf{T}|\hat{\Phi}) \tag{8.78}$$

In a manner similar to the standard HMM, $\gamma_{t,\tau}(i, j)$ can be defined as the transition probability from state i at time t to state j with time duration τ in state j. $\gamma_{t,\tau}(i, j)$ can be written as:

$$\gamma_{t,\tau}(i, j) = \alpha_t(i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t+l}) \beta_{t+\tau}(j) / \sum_{k=1}^{N} \alpha_T(k) \tag{8.79}$$

Similarly, the probability of being in state j at time t with duration τ can be computed as:

$$\gamma_{t,\tau}(j) = \sum_{i} \gamma_{t,\tau}(i, j) \tag{8.80}$$

The re-estimation algorithm can be derived from Eq. (8.80), the Viterbi decoding algorithm can be used for the time duration model, and the optimal path can be determined according to:

$$V_t(j) = \max_{i} \max_{\tau} [V_{t-\tau}(i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t-\tau+l})] \tag{8.81}$$

There are drawbacks to the use of the time duration modeling discussed here. One is the great increase in computational complexity by a factor of $O(D^2)$, where D is the time

duration distribution length. Another problem is the large number of additional parameters D that must be estimated. One proposed remedy is to use a continuous density function instead of the discrete distribution $d_j(\tau)$.

In practice, duration models offer only modest improvement for speaker-independent continuous speech recognition. Many systems even eliminate the transition probability completely because the output probabilities are so dominant. Nevertheless, duration information is very effective for pruning unlikely candidates during the large-vocabulary speech recognition decoding process.

8.5.2. First-Order Assumption

As you can see from the previous section, the duration of each stationary segment captured by a single state is inadequately modeled. Another way to alleviate the duration problem is to eliminate the first-order transition assumption and to make the underlying state sequence a second-order Markov chain [32]. As a result, the transition probability between two states at time t depends on the states in which the process was at time t-1 and t-2. For a given state sequence $\mathbf{S} = \{s_1, s_2, \ldots s_T\}$, the probability of the state should be computed as:

$$P(\mathbf{S}) = \prod_t a_{s_{t-2}s_{t-1}s_t} \tag{8.82}$$

where $a_{s_{t-2}s_{t-1}s_t} = P(s_t | s_{t-2}s_{t-1})$ is the transition probability at time t, given the two-order state history. The re-estimation procedure can be readily extended based on Eq. (8.82). For example, the new forward probability can be re-defined as:

$$\alpha_t(j,k) = P(X_1^t, s_{t-1} = j, s_t = k|\lambda) = \sum_i \alpha_{t-1}(i,j) a_{ijk} b_k(X_t) \tag{8.83}$$

where $a_{ijk} = P(s_t = k | s_{t-2} = i, s_{t-1} = j)$. Similarly, we can define the backward probability as:

$$\beta_t(i,j) = P(X_{t+1}^T | s_{t-1} = i, s_t = j, \lambda) = \sum_k a_{ijk} b_k(X_{t+1}) \beta_{t+1}(j,k) \tag{8.84}$$

With Eq. (8.83) and (8.84), the MLE estimates can be derived easily based on the modified $\gamma_t(i,j,k)$:

$$\begin{aligned} \gamma_t(i,j,k) &= P(s_{t-1} = i, s_t = j, s_{t+1} = k, \mathbf{X}|\Phi) \\ &= \alpha_t(i,j) a_{ijk} b_k(X_{t+1}) \beta_{t+1}(j,k) / P(\mathbf{X}|\Phi) \end{aligned} \tag{8.85}$$

In practice, the second-order model is computationally very expensive as we have to consider the increased state space, which can often be realized with an equivalent first-order hidden Markov model on the two-fold product state space. It has not offered significantly improved accuracy to justify its increase in computational complexity for most applications.

8.5.3. Conditional Independence Assumption

The third major weakness in HMMs is that all observation frames are dependent only on the state that generated them, not on neighboring observation frames. The conditional independence assumption makes it hard to effectively handle nonstationary frames that are strongly correlated. There are a number of ways to alleviate the conditional independence assumption [34]. For example, we can assume the output probability distribution depends not only on the state but also on the previous frame. Thus, the probability of a given state sequence can be rewritten as:

$$P(\mathbf{X}|\mathbf{S},\Phi) = \prod_{t=1}^{T} P(X_t|X_{t-1},s_t,\Phi) \tag{8.86}$$

As the parameter space becomes huge, we often need to quantize X_{t-1} into a smaller set of codewords so that we can keep the number of free parameters under control. Thus, Eq. (8.86) can be simplified as:

$$P(\mathbf{X}|\mathbf{S},\Phi) = \prod_{t=1}^{T} P(X_t|\Re(X_{t-1}),s_t,\Phi) \tag{8.87}$$

where $\Re(\,)$ denotes the quantized vector that has a small codebook size, L. Although this can dramatically reduce the space of the free conditional output probability distributions, the total number of free parameters will still increase by L times.

The re-estimation for conditional dependent HMMs can be derived with the modified Q-function, as discussed in the previous sections. In practice, it has not demonstrated convincing accuracy improvement for large-vocabulary speech recognition.

8.6. Historical Perspective and Further Reading

The Markov chain was named after Russian scientist A. Markov for his pioneering work in analyzing the letter sequence in the text of a literary work in 1913 [33]. In the 1960s, Baum and others further developed efficient methods for training the model parameters [4, 5]. When the observation is real valued, the use of continuous or semi-continuous HMMs can improve the overall performance. Baum et al. also developed the method to use continuous density functions that are strictly log concave [5], which was relaxed by Liporace [31] and expanded by Juang to include mixture density functions [27].

The Viterbi algorithm shares the same concept that was independently discovered by researchers in many separate fields [28], including Vintsyuk [42], Needleman and Wunsch [35], Sankoff [41], Sakoe and Chiba [40], and Wagner and Fischer [44].

Jim Baker did his Ph.D. thesis under Raj Reddy at Carnegie Mellon using HMMs for speech recognition [3]. At the same time Fred Jelinek and his colleagues at IBM Research pioneered widespread applications [23]. Since the 1980s, partly because of the DARPA-

funded speech projects, HMMs have become a mainstream technique for modeling speech, as exemplified by advanced systems developed at BBN, Bell Labs, Carnegie Mellon, IBM, Microsoft, SRI, and others [9, 17, 29, 46]. The Ph.D. theses from Kai-Fu Lee [29], Hsiao-Wuen Hon [16], and Mei-Yuh Hwang [22] at Carnegie Mellon addressed many important practical issues in using HMMs for speech recognition. There are also a number of good books on the practical use of HMMs [18, 24, 38, 45].

The choice of different output probabilities depends on a number of factors such as the availability of training data, the feature characteristics, the computational complexity, and the number of free parameters [19] [34]. The semi-continuous model, also known as the tied-mixture model, was independently proposed by Huang and Jack [21] and Bellegarda and Nahamoo [6]. Other improvements include explicit duration modeling [1, 11, 13, 14, 30, 39], high-order and conditional models [7, 32, 34], which have yet to be shown effective for practical speech recognition.

Both Carnegie Mellon University's open speech software[4] and Cambridge University's HTK[5] are a good starting point for those interested in using the existing tools for running experiments.

HMMs have become the most prominent techniques for speech recognition today. Most of the state-of-the-art speech recognition systems on the market are based on HMMs described in this chapter.

REFERENCES

[1] Anastasakos, A., R. Schwartz, and H. Sun, "Duration Modeling in Large Vocabulary Speech Recognition" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, MI, pp. 628-631.

[2] Bahl, L.R., *et al.*, "Speech Recognition with Continuous-Parameter Hidden Markov Models," *Computer Speech and Language*, 1987, **2**, pp. 219-234.

[3] Baker, J.K., "The DRAGON System—An Overview," *Trans. on Acoustics, Speech and Signal Processing*, 1975, **23**(1), pp. 24-29.

[4] Baum, L.E. and J.A. Eagon, "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology," *Bulletin of American Mathematical Society*, 1967, **73**, pp. 360-363.

[5] Baum, L.E., *et al.*, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Mathematical Statistics*, 1970, **41**, pp. 164-171.

[6] Bellegarda, J.R. and D. Nahamoo, "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, pp. 13-16.

[7] Brown, P., *The Acoustic-Modeling Problem in Automatic Speech Recognition*, Ph.D. Thesis in *Computer Science*, 1987, Carnegie Mellon University, Pittsburgh.

[4] http://www.speech.cs.cmu.edu/sphinx/

[5] http://htk.eng.cam.ac.uk/

[8] Brown, P.F., *et al.*, "The Mathematics of Statistical Machine Translation: Parameter Estimation," *Computational Linguistics*, 1995, **19**(2), pp. 263-312.

[9] Chou, W., C.H. Lee, and B.H. Juang, "Minimum Error Rate Training of Inter-Word Context Dependent Acoustic Model Units in Speech Recognition" in *Proc. of the Int. Conf. on Spoken Language Processing*, 1994, Yokohama, Japan, pp. 439-442.

[10] Church, K., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," *Proc. of the Second Conf. on Applied Natural Language Processing*, 1988, Austin, Texas, pp. 136-143.

[11] Deng, L., M. Lennig, and P. Mermelstein, "Use of Vowel Duration Information in a Large Vocabulary Word Recognizer," *Journal of the Acoustical Society of America*, 1989, **86**(2 August), pp. 540-548.

[12] DeRose, S.J., "Grammatical Category Disambiguation by Statistical Optimization," *Computational Linguistics*, 1988(1), pp. 31-39.

[13] Dumouchel, P. and D. Shaughnessy, "Segmental Duration and HMM Modeling," *Proc. of the European Conf. on Speech Communication and Technology*, 1995, Madrid, Spain, pp. 803-806.

[14] Ferguson, J.D., "Variable Duration Models for Speech" in *Proc. of the Symposium on the Application of Hidden Markov Models to Text and Speech*, J.D. Ferguson, ed., 1980, Princeton, New Jersey, pp. 143-179.

[15] Fiscus, J., "A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER)," *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997, Santa Barbara, CA, pp. 347-352.

[16] Hon, H.W., *Vocabulary-Independent Speech Recognition: The VOCIND System*, Ph.D Thesis in *Department of Computer Science* 1992, Carnegie Mellon University, Pittsburgh.

[17] Huang, X.D., *et al.*, "The SPHINX-II Speech Recognition System: An Overview," *Computer Speech and Language*, 1993, pp. 137-148.

[18] Huang, X.D., Y. Ariki, and M.A. Jack, *Hidden Markov Models for Speech Recognition*, 1990, Edinburgh, U.K., Edinburgh University Press.

[19] Huang, X.D., *et al.*, "A Comparative Study of Discrete, Semicontinuous, and Continuous Hidden Markov Models," *Computer Speech and Language*, 1993, 7(4), pp. 359-368.

[20] Huang, X.D., *et al.*, "Deleted Interpolation and Density Sharing for Continuous Hidden Markov Models," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1996.

[21] Huang, X.D. and M.A. Jack, "Semi-Continuous Hidden Markov Models with Maximum Likelihood Vector Quantization" in *IEEE Workshop on Speech Recognition*, 1988.

[22] Hwang, M., *Subphonetic Modeling in HMM-based Speech Recognition Systems*, Ph.D. Thesis in *Computer Science*, 1994, Carnegie Mellon University, Pittsburgh.

[23] Jelinek, F., "Continuous Speech Recognition by Statistical Methods," *Proc. of the IEEE*, 1976, **64**(4), pp. 532-556.

[24] Jelinek, F., *Statistical Methods for Speech Recognition*, 1998, Cambridge, MA, MIT Press.

[25] Jiang, L. and X. Huang, "Unified Decoding and Feature Representation for Improved Speech Recognition," *Proc. of the 6th European Conf. on Speech Communication and Technology*, 1999, Budapest, Hungary, pp. 1331-1334.

[26] Juang, B.H., W. Chou, and C.H. Lee, "Statistical and Discriminative Methods for Speech Recognition" in *Automatic Speech and Speaker Recognition—Advanced Topics*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds., 1996, Boston, pp. 109-132, Kluwer Academic Publishers.

[27] Juang, B.H., S.E. Levinson, and M.M. Sondhi, "Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains," *IEEE Trans. on Information Theory*, 1986, **IT-32**(2), pp. 307-309.

[28] Kruskal, J., "An Overview of Sequence Comparison" in *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J. Kruskal, eds. 1983, Reading, MA., Addison-Wesley, pp. 1-44.

[29] Lee, K.F., *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*, Ph.D. Thesis in Computer Science Dept., 1988, Carnegie Mellon University, Pittsburgh.

[30] Levinson, S.E., "Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition," *Computer Speech and Language*, 1986, pp. 29-45.

[31] Liporace, L.R., "Maximum Likelihood Estimation for Multivariate Observations of Markov Sources," *IEEE Trans. on Information Theory*, 1982, **28**, pp. 729-734.

[32] Mari, J., J. Haton, and A. Kriouile, "Automatic Word Recognition Based on Second-Order Hidden Markov Models," *IEEE Trans. on Speech and Audio Processing*, 1977, **5**(1), pp. 22-25.

[33] Markov, A.A., "An Example of Statistical Investigation in the Text of 'Eugene Onyegin', Illustrating Coupling of Tests in Chains," *Proc. of the Academy of Sciences of St. Petersburg*, 1913, Russia, pp. 153-162.

[34] Ming, J. and F. Smith, "Modelling of the Interframe Dependence in an HMM Using Conditional Gaussian Mixtures," *Computer Speech and Language*, 1996, **10**(4), pp. 229-247.

[35] Needleman, S. and C. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino-acid Sequence of Two Proteins," *Journal of Molecular Biology*, 1970, **48**, pp. 443-453.

[36] Normandin, Y., "Maximum Mutual Information Estimation of Hidden Markov Models" in *Automatic Speech and Speaker Recognition*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds., 1996, Norwell, MA, Kluwer Academic Publishers.

[37] Rabiner, L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. of IEEE*, 1989, **77**(2), pp. 257-286.

[38] Rabiner, L.R. and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall Signal Processing Series, ed. A.V. Oppenheim, 1993, Englewood Cliffs, NJ, Prentice-Hall.

[39] Russell, M.J. and R.K. Moore, "Explicit Modeling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1985, pp. 5-8.

[40] Sakoe, H. and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1978, **26**(1), pp. 43-49.

[41] Sankoff, D., "Matching Sequences under Deletion-Insertion Constraints," *Proc. of the National Academy of Sciences*, 1972, **69**, pp. 4-6.

[42] Vintsyuk, T.K., "Speech Discrimination by Dynamic Programming," *Cybernetics*, 1968, **4**(1), pp. 52-57.

[43] Viterbi, A.J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. on Information Theory*, 1967, **13**(2), pp. 260-269.

[44] Wagner, R. and M. Fischer, "The String-to-String Correction Problem," *Journal of the ACM*, 1974, **21**, pp. 168-173.

[45] Waibel, A.H. and K.F. Lee, *Readings in Speech Recognition*, 1990, San Mateo, CA, Morgan Kaufman Publishers.

[46] Young, S.J. and P.C. Woodland, "The Use of State Tying in Continuous Speech Recognition," *Proc. of Eurospeech*, 1993, Berlin, pp. 2203-2206.

CHAPTER 9

Acoustic Modeling

After years of research and development, accuracy of automatic speech recognition remains one of the most important research challenges. A number of well-known factors determines accuracy; those most noticeable are variations in context, in speaker, and in environment. Acoustic modeling plays a critical role in improving accuracy and is arguably the central part of any speech recognition system.

For the given acoustic observation $\mathbf{X} = X_1 X_2 ... X_n$, the goal of speech recognition is to find out the corresponding word sequence $\hat{\mathbf{W}} = w_1 w_2 ... w_m$ that has the maximum posterior probability $P(\mathbf{W}|\mathbf{X})$ as expressed by Eq. (9.1).

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{X}) = \arg\max_{\mathbf{W}} \frac{P(\mathbf{W})P(\mathbf{X}|\mathbf{W})}{P(\mathbf{X})} \tag{9.1}$$

Since the maximization of Eq. (9.1) is carried out with the observation $\mathbf{X}$ fixed, the above maximization is equivalent to maximization of the following equation:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W}) \tag{9.2}$$

The practical challenge is how to build accurate acoustic models, $P(\mathbf{X}|\mathbf{W})$, and language models, $P(\mathbf{W})$, that can truly reflect the spoken language to be recognized. For large-vocabulary speech recognition, since there are a large number of words, we need to decompose a word into a subword sequence. Thus $P(\mathbf{X}|\mathbf{W})$ is closely related to phonetic modeling. $P(\mathbf{X}|\mathbf{W})$ should take into account speaker variations, pronunciation variations, environmental variations, and context-dependent phonetic coarticulation variations. Last, but not least, any static acoustic or language model will not meet the needs of real applications. So it is vital to dynamically adapt both $P(\mathbf{W})$ and $P(\mathbf{X}|\mathbf{W})$ to maximize $P(\mathbf{W}|\mathbf{X})$ while using spoken language systems. The decoding process of finding the best word sequence $\mathbf{W}$ to match the input speech signal $\mathbf{X}$ in speech recognition systems is more than a simple pattern recognition problem, since in continuous speech recognition you have an infinite number of word patterns to search, as discussed in detail in Chapters 12 and 13.

In this chapter we focus on discussing solutions that work well in practice. To highlight solutions that are effective, we use the Whisper speech recognition system [49] developed at Microsoft Research as a concrete example to illustrate how to build a working system and how various techniques can help to reduce speech recognition errors.[1] We hope that by studying what worked well in the past we can illuminate the possibilities for further improvement of the state of the art.

The hidden Markov model we discussed in Chapter 8 is the underpinning for acoustic phonetic modeling. It provides a powerful way to integrate segmentation, time warping, pattern matching, and context knowledge in a unified manner. The underlying technologies are undoubtedly evolving, and the research community is aggressively searching for more powerful solutions. Most of the techniques discussed in this chapter can be readily derived from the fundamentals discussed in earlier chapters.

9.1. VARIABILITY IN THE SPEECH SIGNAL

The research community has produced technologies that, with some constraints, can accurately recognize spoken input. Admittedly, today's state-of-the-art systems still cannot match humans' performance. Although we can build a very accurate speech recognizer for a particular speaker, in a particular language and speaking style, in a particular environment, and limited to a particular task, it remains a research challenge to build a recognizer that can essentially understand anyone's speech, in any language, on any topic, in any free-flowing style, and in almost any speaking environment.

[1] Most of the experimental results used here are based on a development test set for the 60,000-word speaker-independent continuous dictation task. The training set consists of 35,000 utterances from about 300 speakers. The test set consists of 410 utterances from 10 speakers that were not used in the training data. The language model is derived from 2 billion words of English text corpora.

Accuracy and robustness are the ultimate measures for the success of speech recognition algorithms. There are many reasons why existing algorithms or systems did not deliver what people want. In the sections that follow we summarize the major factors involved.

9.1.1. Context Variability

Spoken language interaction between people requires knowledge of word meanings, communication context, and common sense. Words with widely different meanings and usage patterns may have the same phonetic realization. Consider the challenge represented by the following utterance:

> *Mr. Wright should write to Ms. Wright right away about his Ford or four door Honda.*

For a given word with the same pronunciation, the meaning could be dramatically different, as indicated by *Wright*, *write*, and *right*. What makes it even more difficult is that *Ford or* and *Four Door* are not only phonetically identical, but also semantically relevant. The interpretation is made within a given word boundary. Even with smart linguistic and semantic information, it is still impossible to decipher the correct word sequence, unless the speaker pauses between words or uses intonation to set apart these semantically confusable phrases.

In addition to the context variability at word and sentence level, you can find dramatic context variability at the phonetic level. As illustrated in Figure 9.1, the acoustic realization of phoneme */eel* for word *peat* and *wheel* depends on its left and right context. The dependency becomes more important in fast speech or spontaneous speech conversations, since many phonemes are not fully realized.

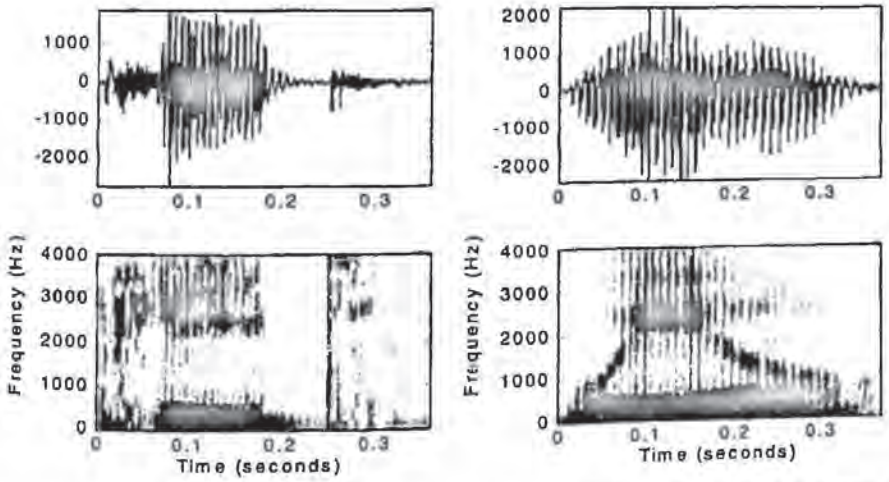


Figure 9.1 Waveforms and spectrograms for words *peat* (left) and *wheel* (right). The phoneme */ee/* is illustrated with two different left and right contexts. This illustrates that different contexts may have different effects on a phone.

9.1.2. Style Variability

To deal with acoustic realization variability, a number of constraints can be imposed on the use of the speech recognizer. For example, we can have an *isolated* speech recognition system, in which users have to pause between each word. Because the pause provides a clear boundary for the word, we can easily eliminate errors such as *Ford or* and *four door*. In addition, isolated speech provides a correct silence context to each word so that it is easier to model and decode the speech, leading to a significant reduction in computational complexity and error rate. In practice, the word-recognition error rate of an isolated speech recognizer can typically be reduced by more than a factor of three (from 7% to 2%) as compared with to a comparable continuous speech recognition system [5]. The disadvantage is that such an isolated speech recognizer is unnatural to most people. The throughput is also significantly lower than that for continuous speech.

In continuous speech recognition, the error rate for casual, spontaneous speech, as occurs in our daily conversation, is much higher than for carefully articulated read-aloud speech. The rate of speech also affects the word recognition rate. It is typical that the higher the *speaking rate* (words/minute), the higher the error rate. If a person whispers, or shouts, to reflect his or her emotional changes, the variation increases even more significantly.

9.1.3. Speaker Variability

Every individual speaker is different. The speech he or she produces reflects the physical vocal tract size, length and width of the neck, a range of physical characteristics, age, sex, dialect, health, education, and personal style. As such, one person's speech patterns can be entirely different from those of another person. Even if we exclude these interspeaker differences, the same speaker is often unable to precisely produce the same utterance. Thus, the shape of the vocal tract movement and rate of delivery may vary from utterance to utterance, even with dedicated effort to minimize the variability.

For *speaker-independent* speech recognition, we typically use more than 500 speakers to build a combined model. Such an approach exhibits large performance fluctuations among new speakers because of possible mismatches in the training data between exiting speakers and new ones [50]. In particular, speakers with accents have a tangible error-rate increase of 2 to 3 times.

To improve the performance of a speaker-independent speech recognizer, a number of constraints can be imposed on its use. For example, we can have a user enrollment that requires the user to speak for about 30 minutes. With the *speaker-dependent* data and training, we may be able to capture various speaker-dependent acoustic characteristics that can significantly improve the speech recognizer's performance. In practice, speaker-dependent speech recognition offers not only improved accuracy but also improved speed, since decoding can be more efficient with an accurate acoustic and phonetic model. A typical speaker-dependent speech recognition system can reduce the word recognition error by more than 30% as compared with a comparable speaker-independent speech recognition system.

The disadvantage of speaker-dependent speech recognition is that it takes time to collect speaker-dependent data, which may be impractical for some applications such as an automatic telephone operator. Many applications have to support walk-in speakers, so speaker-independent speech recognition remains an important feature. When the amount of speaker-dependent data is limited, it is important to make use of both speaker-dependent and speaker-independent data using *speaker-adaptive* training techniques, as discussed in Section 9.6. Even for speaker-independent speech recognition, you can still use speaker-adaptive training based on recognition results to quickly adapt to each individual speaker during usage.

9.1.4. Environment Variability

The world we live in is full of sounds of varying loudness from different sources. When we interact with computers, we may have people speaking in the background. Someone may slam the door, or the air conditioning may start humming without notice. If speech recognition is embedded in mobile devices, such as PDAs (personal digital assistants) or cellular phones, the spectrum of noises varies significantly because the owner moves around. These external parameters, such as the characteristics of the environmental noise and the type and placement of the microphone, can greatly affect speech recognition system performance. In addition to the background noises, we have to deal with noises made by speakers, such as lip smacks and noncommunication words. Noise may also be present from the input device itself, such as microphone and A/D interference noises.

In a similar manner to speaker-independent training, we can build a system by using a large amount of data collected from a number of environments; this is referred to as *multistyle training* [70]. We can use adaptive techniques to normalize the mismatch across different environment conditions in a manner similar to speaker-adaptive training, as discussed in Chapter 10. Despite the progress being made in the field, environment variability remains as one of the most severe challenges facing today's state-of-the-art speech systems.

9.2. How to Measure Speech Recognition Errors

It is critical to evaluate the performance of speech recognition systems. The word recognition error rate is widely used as one of the most important measures. When you compare different acoustic modeling algorithms, it is important to compare their relative error reduction. Empirically, you need to have a test data set that contains more than 500 sentences (with 6 to 10 words for each sentence) from 5 to 10 different speakers to reliably estimate the recognition error rate. Typically, you need to have more than 10% relative error reduction to consider adopting a new algorithm.

As a sanity check, you may want to use a small sample from the training data to measure the performance of the *training set*, which is often much better than what you can get from testing new data. Training-set performance is useful in the development stage to identify potential implementation bugs. Eventually, you need to use a *development set* that typically consists of data never used in training. Since you may tune a number of parameters

with your development set, it is important to evaluate performance of a *test set* after you decide the optimal parameter setting. The test set should be completely new with respect to both training and parameter tuning.

There are typically three types of word recognition errors in speech recognition:

- *Substitution*: an incorrect word was substituted for the correct word
- *Deletion*: a correct word was omitted in the recognized sentence
- *Insertion*: an extra word was added in the recognized sentence[2]

For instance, a speech recognition system may produce an incorrect result as follows, where substitutions are ***bold***, insertions are underlined, and deletions are denoted as **. There are four errors in this example.

> **Correct:** *Did mob mission area of the Copeland ever go to m4 in nineteen eighty one*
> **Recognized:** *Did mob mission area ** **the copy** land ever go to m4 in nineteen **east** one*

To determine the minimum error rate, you can't simply compare two word sequences one by one. For example, suppose you have utterance *The effect is clear* recognized as *Effect is not clear*. If you compare word to word, the error rate is 75% (*The* vs. *Effect*, *effect* vs. *is*, *is* vs. *not*). In fact, the error rate is only 50% with one deletion (*The*) and one insertion (*not*). In general, you need to align a recognized word string against the correct word string and compute the number of substitutions (*Subs*), deletions (*Dels*), and insertions (*Ins*). The *Word Error Rate* is defined as:

$$\text{Word Error Rate} = 100\% \times \frac{Subs + Dels + Ins}{\text{No. of words in the correct sentenc}} \tag{9.3}$$

This alignment is also known as the *maximum substring matching* problem, which can be easily handled by the dynamic programming algorithm discussed in Chapter 8.

Let the correct word string be $w_1 w_2 \cdots w_n$, where w_i denotes the ith word in the correct word string, and the recognized word string be $\hat{w}_1 \hat{w}_2 \cdots \hat{w}_m$, where $\hat{w}_i$ denotes the i^{th} word in the recognized word string. We denote $R[i, j]$ as the minimum error of aligning substring $w_1 w_2 \cdots w_n$ against substring $\hat{w}_1 \hat{w}_2 \cdots \hat{w}_m$. The optimal alignment and the associated word error rate $R[n, m]$ for correct word string $w_1 w_2 \cdots w_n$ and the recognized word string $\hat{w}_1 \hat{w}_2 \cdots \hat{w}_m$ are obtained via the dynamic programming algorithm illustrated in Algorithm 9.1. The accumulated cost function $R[i, j]$ progresses from $R[1, 1]$ to $R[n, m]$ corresponding to the minimum distance from (1, 1) to (n, m). We store the back pointer information $B[i, j]$ as we move along. When we reach the final grid (n, m), we back trace along the optimal path to find out if there are substitutions, deletions, or insertions on the matched path, as stored in $B[i, j]$.

[2] Even for isolated speech recognition, you may still have the insertion error, since the word boundary needs to be detected in most applications. It is possible that one isolated utterance is recognized as two words.

ALGORITHM 9.1: ALGORITHM TO MEASURE THE WORD ERROR RATE

Step 1: *Initialization* $R[0,0] = 0$ $R[i,j] = \infty$ if $(i < 0)$ or $(j < 0)$ $B[0,0] = 0$

Step 2: *Iteration*

for $i = 1,\ldots,n$ {

for $j = 1,\ldots,m$ {

$$R[i,j] = \min \begin{bmatrix} R[i-1,j]+1 \text{ (deletion)} \\ R[i-1,j-1] \text{ (match)} \\ R[i-1,j-1]+1 \text{ (substitution)} \\ R[i,j-1]+1 \text{ (insertion)} \end{bmatrix}$$

$$B[i,j] = \begin{cases} 1 & \text{if deletion} \\ 2 & \text{if insertion} \\ 3 & \text{if match} \\ 4 & \text{if substitution} \end{cases} \quad \} \}$$

Step 3: *Backtracking and termination*

$$\text{word error rate} = 100\% \times \frac{R(n,m)}{n}$$

$$\text{optimal backward path} = (s_1, s_2, \ldots, 0)$$

$$\text{where } s_1 = B[n,m],\ s_t = \begin{bmatrix} B[i-1,j] \text{ if } s_{t-1} = 1 \\ B[i,j-1] \text{ if } s_{t-1} = 2 \\ B[i-1,j-1] \text{ if } s_{t-1} = 3 \text{ or } 4 \end{bmatrix} \text{ for } t = 2,\ldots \text{ until } s_t = 0$$

For applications involved with rejection, such as word confidence measures as discussed in Section 9.7, you need to measure both false rejection rate and false acceptance rate. In speaker or command verification, the false acceptance of a valid user/command is also referred to as Type I error, as opposed to the false rejection of a valid user/command (Type II) [17]. A higher false rejection rate generally leads to a lower false acceptance rate. A plot of the false rejection rate versus the false acceptance rate, widely used in communication theory, is called the *receiver operating characteristic* (ROC) curve.

9.3. SIGNAL PROCESSING—EXTRACTING FEATURES

The role of a signal processing module, as illustrated in Figure 1.2, is to reduce the data rate, to remove noises, and to extract salient features that are useful for subsequent acoustic matching. Using as building blocks the topics we discussed in earlier chapters, we briefly illustrate here what is important in modeling speech to deal with variations we must address. More advanced environment normalization techniques are discussed in Chapter 10.

9.3.1. Signal Acquisition

Today's computers can handle most of the necessary speech signal acquisition tasks in software. For example, most PC sound cards have direct memory access, and the speech can be digitized to memory without burdening the CPU with input/output interrupts. The operating system can correctly handle most of the necessary AD/DA functions in real time.

To perform speech recognition, a number of components—such as digitizing speech, feature extraction and transformation, acoustic matching, and language model-based search—can be pipelined time-synchronously from left to right. Most operating systems can supply mechanisms for organizing pipelined programs in a multitasking environment. Buffers must be appropriately allocated so that you can ensure time-synchronous processing of each component. Large buffers are generally required on slow machines because of potential delays in processing an individual component. The right buffer size can be easily determined by experimentally tuning the system with different machine load situations to find a balance between resource use and relative delay.

For speech signal acquisition, the needed buffer typically ranges from 4 to 64 kB with 16-kHz sampling rate and 16-bit A/D precision. In practice, 16-kHz sampling rate is sufficient for the speech bandwidth (8 kHz). Reduced bandwidth, such as telephone channel, generally increases speech recognition error rate. Table 9.1 shows some empirical relative word recognition error increase using a number of different sampling rates. If we take the 8-kHz sampling as our baseline, we can reduce the word recognition error with a comparable recognizer by about 10% if we increase the sampling rate to 11 kHz. If we further increase the sampling rate to 16 kHz, the word recognition error rate can be further reduced by an additional 10%. Further increasing the sampling rate to 22 kHz does not have any additional impact on the word recognition errors, because most of the salient speech features are within an 8-kHz bandwidth.

Table 9.1 Relative error rate reduction with different sampling rates. The reduction is relative to that of the preceding row.

Sampling Rate	Relative Error-Rate Reduction
8 kHz	Baseline
11 kHz	+10%
16 kHz	+10%
22 kHz	+0%

9.3.2. End-Point Detection

To activate speech signal capture, you can use a number of modes including either *push to talk* or *continuously listening*. The push-to-talk mode uses a special push event to activate or

deactivate speech capture, which is immune to the potential background noise and can eliminate unnecessary use of processing resources to detect speech events. This mode sometimes also requires you to *push and hold while talking*. You push to indicate speech's beginning and then release to indicate the end of speech capture. The disadvantage is the necessity to activate the application each time the person speaks.

The continuously listening model listens all the time and automatically detects whether there is a speech signal or not. It needs a so-called *speech end-point detector*, which is typically based on an extremely efficient two-class pattern classifier. Such a classifier is used to filter out obvious silence, but the ultimate decision on the utterance boundary is left to the speech recognizer. In comparison to the push-to-talk mode, the continuously listening mode requires more processing resources, also with potential classification errors.

The endpoint detector is often based on an energy threshold that is a function of time. The logarithm of the energy threshold can be dynamically generated based on energy profiles across a certain period of time. Constraints on word duration can also be imposed to better classify a sequence of frames so that extremely short spikes can be eliminated.

It is not critical for the automatic end-point detector to offer exact end-point accuracy. The key feature required of it is a low rejection rate (i.e., the automatic end-point detector should not interpret speech segments as silence/noise segments). Any false rejection leads to an error in the speech recognizer. On the other hand, a possible false acceptance (i.e., the automatic end-point detector interprets noise segments as speech segments) may be rescued by the speech recognizer later if the recognizer has appropriate noise models, such as specific models for clicks, lip smacks, and background noise.

Explicit end-point detectors work reasonably well with recordings exhibiting a signal-to-noise ratio of 30 dB or greater, but they fail considerably on noisier speech. As discussed, speech recognizers can be used to determine the end points by aligning the vocabulary words preceded and followed by a silence/noise model. This scheme is generally much more reliable than any threshold-based explicit end-point detection, because recognition can jointly detect both the end points and words or other explicit noise classes, but requires more computational resources. A compromise is to use a simple adaptive two-class (speech vs. silence/noise) classifier to locate speech activities (with enough buffers at both ends) and notify the speech recognizer for subsequent processing. For the two-class classifier, we can use both the log-energy and delta log-energy as the feature. Two Gaussian density functions, $\{\Phi_1, \Phi_2\} = \Phi$, can be used to model the background stationary noise and speech, respectively. The parameters of the Gaussian density can be estimated using the labeled speech and noise data or estimated in an unsupervised manner.

When enough frames are classified as speech segments by the efficient two-class classifier, the speech recognizer is notified to start recognizing the signal. As shown in Figure 9.2, we should include enough frames before the beginning frame, t_b, for the speech recognizer to minimize the possible detection error. In the same manner, when enough noise/silence frames are detected at t_e, we should keep providing the speech recognizer with enough frames for processing before declaring that the end of the utterance has been reached.



Figure 9.2 End-point detection boundary t_b and t_e may need extra buffering for subsequent speech recognition.

Since there are only two classes, these parameters can be dynamically adapted using the EM algorithm during runtime. As discussed in Chapter 4, the EM algorithm can iteratively estimate the Gaussian parameters without having a precise segmentation between speech and noise segments. This is very important, because we need to keep the parameters dynamic for robust end-point detection in constantly changing environments.

To track the varying background noises, we use an exponential window to give weight to the most recent signal:

$$w_k = \exp(-\alpha k) \tag{9.4}$$

where α is a constant that controls the adaptation rate, and k is the index of the time. In fact, you could use different rates for noise and speech when you use the EM algorithm to estimate the two-class Gaussian parameters. It is advantageous to use a smaller time constant for noise than for speech. With such a weighting window, the means of the Gaussian density, as discussed in Chapter 4, can be rewritten as:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=-\infty}^{t} w_i \dfrac{c_k P(\mathbf{x}_i \mid \Phi_k)\mathbf{x}_i}{\sum_{k=1}^{2} P(\mathbf{x}_i \mid \Phi_k)}}{\sum_{i=-\infty}^{t} w_i \dfrac{c_k P(\mathbf{x}_i \mid \Phi_k)}{\sum_{k=1}^{2} P(\mathbf{x}_i \mid \Phi_k)}}, k \in \{0,1\} \tag{9.5}$$

9.3.3. MFCC and Its Dynamic Features

The extraction of reliable features is one of the most important issues in speech recognition. There are a large number of features we can use. However, as discussed in Chapter 4, the

curse-of-dimensionality problem reminds us that the amount of training data is always limited. Therefore, incorporation of additional features may not lead to any measurable error reduction. This does not necessarily mean that the additional features are poor ones, but rather that we may have insufficient data to reliably model those features.

The first feature we use is the speech waveform itself. In general, time-domain features are much less accurate than frequency-domain features such as the mel-frequency cepstral coefficients (MFCC) discussed in Chapter 6 [23]. This is because many features such as formants, useful in discriminating vowels, are better characterized in the frequency domain with a low-dimension feature vector.

As discussed in Chapter 2, temporal changes in the spectra play an important role in human perception. One way to capture this information is to use *delta coefficients* that measure the change in coefficients over time. Temporal information is particularly complementary to HMMs, since HMMs assume each frame is independent of the past, and these dynamic features broaden the scope of a frame. It is also easy to incorporate new features by augmenting the static feature.

When 16-kHz sampling rate is used, a typical state-of-the-art speech system can be build based on the following features.

- 13th-order MFCC $\mathbf{c}_k$
- 13th-order 40-msec 1st-order delta MFCC computed from $\Delta\mathbf{c}_k = \mathbf{c}_{k+2} - \mathbf{c}_{k-2}$
- 13th-order 2nd-order delta MFCC computed from $\Delta\Delta\mathbf{c}_k = \Delta\mathbf{c}_{k+1} - \Delta\mathbf{c}_{k-1}$

A short-time analysis Hamming window of 25 ms is typically used to compute the MFCC $\mathbf{c}_k$. The window shift is typically 10 ms. Please note that $c_k[0]$ is included in the feature vector, which has a role similar to that of the log power. The feature vector used for speech recognition is typically a combination of these features

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{c}_k \\ \Delta\mathbf{c}_k \\ \Delta\Delta\mathbf{c}_k \end{pmatrix} \tag{9.6}$$

The relative error reduction with a typical speech recognition system is illustrated in Table 9.2. As you can see from the table, the 13th-order MFCC outperforms 13th-order LPC cepstrum coefficients, which indicates that perceptually motivated mel-scale representation indeed helps recognition. In a similar manner, perceptually based LPC features such as PLP can achieve similar improvement. The MFCC order has also been studied experimentally for speech recognition. The higher-order MFCC does not further reduce the error rate in comparison with the 13th-order MFCC, which indicates that the first 13 coefficients already contain most salient information needed for speech recognition. In addition to mel-scale representation, another perceptually motivated feature such as the first- and second-order delta features can significantly reduce the word recognition error, while the higher-order delta features provide no further information.

Feature extraction in these experiments is typically optimized together with the classifier, since there are a number of modeling assumptions, such as the diagonal covariance in the Gaussian density function, that are closely related to what features to use. It is possible that these relative error reductions would vary if a different speech recognizer were used.

Table 9.2 Relative error reduction with different features. The reduction is relative to that of the preceding row.

Feature Set	Relative Error Reduction
13th-order LPC cepstrum coefficients	Baseline
13th-order MFCC	+10%
16th-order MFCC	+0%
+1st- and 2nd-order dynamic features	+20%
+3rd-order dynamic features	+0%

9.3.4. Feature Transformation

Before you use feature vectors such as MFCC for recognition, you can preprocess or transform them into a new space that alleviates environment noise, channel distortion, and speaker variations. You can also transform the features that are most effective for preserving class separability so that you can further reduce the recognition error rate. Since we devote Chapter 10 completely to environment and channel normalization, we briefly discuss here how we can transform the feature vectors to improve class separability.

To further reduce the dimension of the feature vector, you can use a number of dimension reduction techniques to map the feature vector into more effective representations. If the mapping is linear, the mapping function is well defined and you can find the coefficients of the linear function so as to optimize your objective functions. For example, when you combine the first- and second-order dynamic features with the static MFCC vector, you can use *principal-component analysis* (PCA) (also known as *Karhunen-Loeve* transform) [32] to map the combined feature vector into a smaller dimensional vector. The optimum basis vectors of the principal-component analysis are the eigenvectors of the covariance matrix of a given distribution. In practice, you can compute the eigenvectors of the autocorrelation matrix as the basis vectors. The effectiveness of the transformed vector, in terms of representing the original feature vector, is determined by the corresponding eigenvalue of each value in the vector. You can discard the feature with the smallest eigenvalue, since the mean-square error between the transformed vector and the original vector is determined by the eigenvalue of each feature in the vector. In addition, the transformed feature vector is uncorrelated. This is particularly suitable for the Gaussian probability density function with a diagonal covariance matrix.

The recognition error is the best criterion for deciding what feature sets to use. However, it is hard to obtain such an estimate to evaluate feature sets systematically. A simpler

approach is to use within-class and between-class scatter matrices to formulate criteria of class separability, which is also called *Linear Discriminant Analysis* (LDA) transformation. We can compute the within-class scatter matrix as:

$$S_w = \sum_{\mathbf{x}\in\omega_i} P(\omega_i)E\{(\mathbf{x}-\boldsymbol{\mu}_i)(\mathbf{x}-\boldsymbol{\mu}_i)' \mid \omega_i\} = \sum_{\mathbf{x}\in\omega_i} P(\omega_i)\Sigma_i \tag{9.7}$$

where the sum is for all the data $\mathbf{x}$ within the class ω_i. This is the scatter of samples around their respective class mean. On the other hand, the between-class scatter matrix is the scatter of the expected vectors around the mixture mean:

$$S_B = \sum_{\boldsymbol{\mu}_i\in\omega_i} P(\omega_i)(\boldsymbol{\mu}_i - \mathbf{m}_0)(\boldsymbol{\mu}_i - \mathbf{m}_0)' \tag{9.8}$$

where $\mathbf{m}_0$ represents the expected mean vector of the mixture distribution:

$$\mathbf{m}_0 = E\{\mathbf{x}\} = \sum_{\omega_i} P(\omega_i)\mathbf{m}_i \tag{9.9}$$

To formulate criteria to transform feature vector $\mathbf{x}$, we need to derive the linear transformation matrix $\mathbf{A}$. One of the measures can be the trace of $S_w^{-1}S_B$:

$$J = tr(S_w^{-1}S_B) \tag{9.10}$$

The trace is the sum of the eigenvalues of $S_w^{-1}S_B$ and hence the sum of the variances in the principal directions. The number is larger when the between-class scatter is large or the within-class scatter is small. You can derive the transformation matrix based on the eigenvectors of $S_w^{-1}S_B$. In a manner similar to PCA, you can reduce the dimension of the original input feature vector by discarding the smallest eigenvalues [16, 54].

Researchers have used the LDA method to measure the effectiveness of several feature vectors for speaker normalization [41]. Other feature processing techniques designed for speaker normalization include neural-network-based speaker mapping [51], frequency warping for vocal tract normalization (VTN) via mel-frequency scaling [67, 100], and bilinear transformation [2].

To reduce interspeaker variability by a speaker-specific frequency warping, you can simply shift the center frequencies of the mel-spaced filter bank. Let $k\Delta f_{mel}$, $k = 1, \ldots, K$, denote the center frequencies in mel-scale. Then the center frequencies in hertz for a warping factor of α are computed by Eq. (9.11) before the cosine transformation of the MFCC feature vector.

$$f_{Hz}^{\alpha}(k\Delta f_{mel}) = 700(10^{k\Delta f_{mel}/2595} - 1)/\alpha \tag{9.11}$$

The warping factor is estimated for each speaker by computing the likelihood of the training data for feature sets obtained with different warping factors using the HMM. The relative error reduction based on the feature transformation method has been limited, typically under 10%.

9.4. PHONETIC MODELING—SELECTING APPROPRIATE UNITS

As discussed in Chapter 2, the phonetic system is related to a particular language. We focus our discussion on language-independent technologies but use English in our examples to illustrate how we can use the language-independent technologies to model the salient phonetic information in the language. For general-purpose large-vocabulary speech recognition, it is difficult to build whole-word models because:

- Every new task contains novel words without any available training data, such as proper nouns and newly invented jargons.
- There are simply too many words, and these different words may have different acoustic realizations, as illustrated in Chapter 2. It is unlikely that we have sufficient repetitions of these words to build context-dependent word models.

How to select the most basic units to represent salient acoustic and phonetic information for the language is an important issue in designing a workable system. At a high level, there are a number of issues we must consider in choosing appropriate modeling units.

- The unit should be *accurate*, to represent the acoustic realization that appears in different contexts.
- The unit should be *trainable*. We should have enough data to estimate the parameters of the unit. Although words are accurate and representative, they are the least trainable choice in building a working system, since it is nearly impossible to get several hundred repetitions for all the words, unless we are using a speech recognizer that is domain specific, such as a recognizer designed for digits only.
- The unit should be *generalizable*, so that any new word can be derived from a predefined unit inventory for task-independent speech recognition. If we have a fixed set of word models, there is no obvious way for us to derive the new word model.

A practical challenge is how to balance these selection criteria for speech recognition. In this section we compare a number of units and point out their strengths and weaknesses in practical applications.

9.4.1. Comparison of Different Units

What is a unit of language? In English, words are typically considered as a principal carrier of meaning and are seen as the smallest unit that is capable of independent use. As the most natural unit of speech. whole-word models have been widely used for many speech recognition systems. A distinctive advantage of using word models is that we can capture the phonetic coarticulation inherent within these words. When the vocabulary is small, we can create word models that are context dependent.

For example, if the vocabulary is English digits, we can have different word models for the word *one* to represent the word in different contexts. Thus each word model is dependent on its left and right context. If someone says *three one two*, the recognizer uses the word model *one* that specifically depends on the left context *three* and right context *two*. Since the vocabulary is small (10), we need to have only 10*10*10=1000 word models, which is achievable when you collect enough training data. With context-dependent, or even context-independent, word models, a wide range of phonological variations can be automatically accommodated. When these word models are adequately trained, they usually yield the best recognition performance in comparison to other modeling units. Therefore, for small vocabulary recognition, whole-word models are widely used, since they are both *accurate* and *trainable*, and there is no need to be *generalizable*.

While words are suitable units for small-vocabulary speech recognition, they are not a practical choice for large-vocabulary continuous speech recognition. First, each word has to be treated individually, and data cannot be shared across word models; this implies a prohibitively large amount of training data and storage. Second, for some task configurations, the recognition vocabulary may consist of words that never appeared in the training data. As a result, some form of word-model composition technique is required to generate word models. Third, it is very expensive to model interword coarticulation effects or adapt a word-based system for a new speaker, a new channel, or new context usage.

To summarize, word models are *accurate* if enough data are available. Thus, they are *trainable* only for small tasks. They are typically not *generalizable*.

Alternatively, there are only about 50 phones in English, and they can be sufficiently trained with just a few hundred sentences. Unlike word models, phonetic models provide no training problem. Moreover, they are also vocabulary independent by nature and can be trained on one task and tested on another. Thus, phones are more *trainable* and *generalizable*. However, the phonetic model is inadequate because it assumes that a phoneme in any context is identical. Although we may try to say each word as a concatenated sequence of independent phonemes, these phonemes are not produced independently, because our articulators cannot move instantaneously from one position to another. Thus, the realization of a phoneme is strongly affected by its immediately neighboring phonemes. For example, if context-independent phonetic models are used, the same model for *t* must capture various events, such as flapping, unreleased stops, and realizations in */t s/* and */t r/*. Then, if */t s/* is the only context in which *t* occurs in the training, while */t r/* is the only context in the testing,

the model used is highly inappropriate. While word models are not generalizable, phonetic models overgeneralize and, thus, lead to less accurate models.

A compromise between the word and phonetic model is to use larger units such as *syllables*. These units encompass phone clusters that contain the most variable contextual effects. However, while the central portions of these units have no contextual dependencies, the beginning and ending portions are still susceptible to some contextual effects. There are only about 1200 tone-dependent syllables in Chinese and approximately 50 syllables in Japanese, which makes syllable a suitable unit for these languages. Unfortunately, the large number of syllables (over 30,000) in English presents a challenge in terms of trainability.

9.4.2. Context Dependency

If we make units context dependent, we can significantly improve the recognition accuracy, provided there are enough training data to estimate these context-dependent parameters. Context-dependent phonemes have been widely used for large-vocabulary speech recognition, thanks to its significantly improved accuracy and trainability. A context usually refers to the immediate left and/or right neighboring phones.

A *triphone* model is a phonetic model that takes into consideration both the left and the right neighboring phones. If two phones have the same identity but different left or right contexts, they are considered different triphones. We call different realizations of a phoneme *allophones*. Triphones are an example of allophones.

The left and right contexts used in triphones, while important, are only two of many important contributing factors that affect the realization of a phone. Triphone models are powerful because they capture the most important coarticulatory effects. They are generally much more consistent than context-independent phone models. However, as context-dependent models generally have increased parameters, trainability becomes a challenging issue. We need to balance trainability and accuracy with a number of parameter-sharing techniques.

Modeling interword context-dependent phones is complicated. For example, in the word *speech*, pronounced */s p iy ch/*, both left and right contexts for */p/* and */iy/* are known, while the left context for */s/* and the right context for */ch/* are dependent on the preceding and following words in actual sentences. The juncture effect on word boundaries is one of the most serious coarticulation phenomena in continuous speech, especially with short function words like *the* or *a*. Even with the same left and right context identities, there may be significantly different realizations for a phone at different word positions (the *beginning, middle,* or *end* of a word). For example, the phone */t/* in *that rock* is almost extinct, while the phone */t/* in the middle of *theatrical* sounds like */ch/*. This implies that different word positions have an effect on the realization of the same triphone.

In addition to the context, stress also plays an important role in the realization of a particular phone. Stressed vowels tend to have longer duration, higher pitch, and more intensity, while unstressed vowels appear to move toward a neutral, central *schwa*-like phoneme. Agreement about the phonetic identity of a syllable has been reported to be greater in

stressed syllables for both humans and automatic phone recognizers. In English, *word-level* stress is referred to as *free stress*, because the stressed syllable can take on any position within a word, in contrast to *bound stress* found in languages such as French and Polish, where the position of the stressed syllable is fixed within a word. Therefore, stress in English can be used as a constraint for lexical access. In fact, stress can be used as a unique feature to distinguish a set of word pairs, such as ***import*** vs. ***import***, and ***export*** vs. ***export***. For example, the phone set used for Whisper, such as */er/-/axr/* and */ah/-/ix/-/ax/*, describes these stressed and unstressed vowels. One example illustrating how stress can significantly affect the realization of phone is demonstrated in Figure 9.3, where phone */t/* in word *Italy* vs. *Italian* is pronounced differently in American English due the location of the stress, albeit the triphone context is identical for both words.

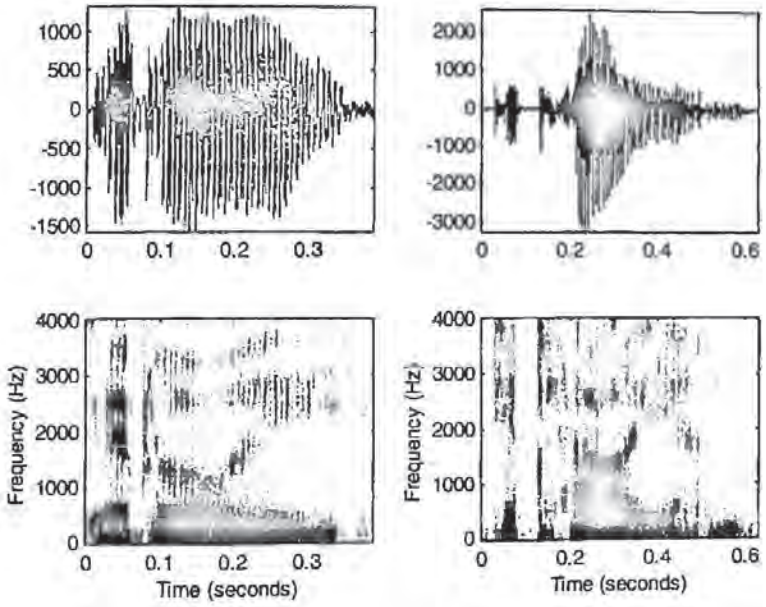


Figure 9.3 The importance of stress is illustrated in *Italy* vs. *Italian* for phone */t/*. The realizations are quite different, even though they share the same left and right context.

Sentence-level stress, on the other hand, represents the overall stress pattern of continuous speech. While sentence-level stress does not change the meaning of any particular lexicon item, it usually increases the relative prominence of portions of the utterance for the purpose of contrast or emphasis. *Contrastive stress* is normally used to coordinate constructions such as *there are **import** records and there are **domestic** ones*, as well as for the purpose of correction, as in *I said **import**, not **export**. Emphatic* stress is commonly used to distinguish a sentence from its negation, e.g., *I **did** have dinner*. Sentence-level stress is very hard to model without incorporating high-level semantic and pragmatic knowledge. In most state-of- the-art speech recognition systems, only word-level stress is used for creating allophones.

9.4.3. Clustered Acoustic-Phonetic Units

Triphone modeling assumes that every triphone context is different. Actually, many phones have similar effects on the neighboring phones. The position of our articulators has an important effect on how we pronounce neighboring vowels. For example, /b/ and /p/ are both labial stops and have similar effects on the following vowel, while /r/ and /w/ are both liquids and have similar effects on the following vowel. Contrary to what we illustrate in Figure 9.1, Figure 9.4 illustrates this phenomenon. It is desirable to find instances of similar contexts and merge them. This would lead to a much more manageable number of models that can be better trained.

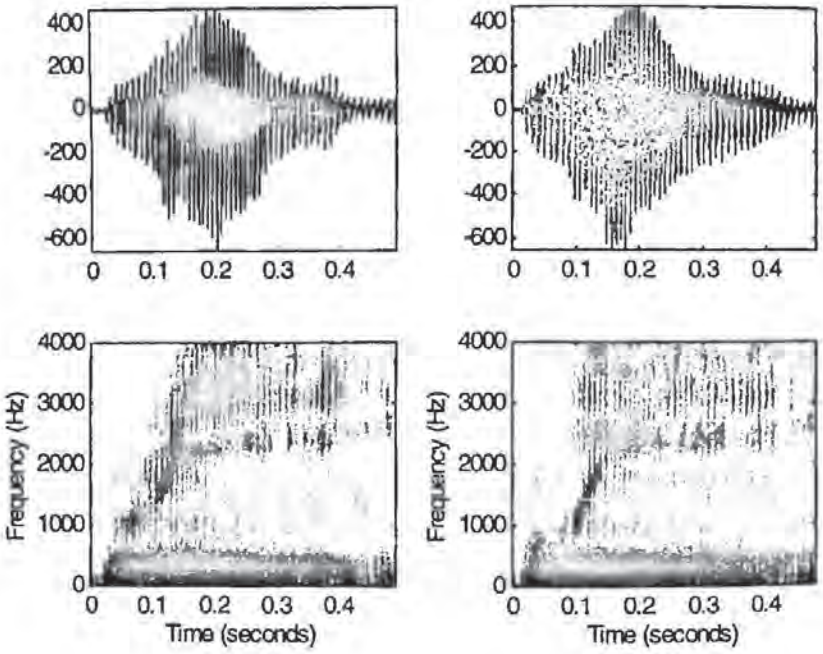


Figure 9.4 The spectrograms for the phoneme /iy/ with two different *left-contexts* are illustrated. Note that /r/ and /w/ have similar effects on /iy/. This illustrates that different left-contexts may have similar effects on a phone.

The trainability and accuracy balance between phonetic and word models can be generalized further to model subphonetic events. In fact, both phonetic and subphonetic units have the same benefits, as they share parameters at the unit level. This is the key benefit in comparison to the word units. Papers by [11, 45, 57, 66, 111] provide examples of the application of this concept to cluster hidden Markov models. For subphonetic modeling, we can treat the state in phonetic HMMs as the basic subphonetic unit. Hwang and Huang further generalized clustering to the state-dependent output distributions across different phonetic models [57]. Each cluster thus represents a set of similar Markov states and is called a

senone [56]. A subword model is thus composed of a sequence of senones after the clustering is finished. The optimal number of senones for a system is mainly determined by the available training corpus and can be tuned on a development set.

Each allophone model is an HMM made of states, transitions, and probability distributions. To improve the reliability of the statistical parameters of these models, some distributions can be tied. For example, distributions for the central portion of an allophone may be tied together to reflect the fact that they represent the stable (context-independent) physical realization of the central part of the phoneme, uttered with a stationary configuration of the vocal tract. Clustering at the granularity of the state rather than the entire model can keep the dissimilar states of two models apart while the other corresponding states are merged, thus leading to better parameter sharing.

Figure 9.5 illustrates how state-based clustering can lead to improved representations. These two HMMs come from the same phone class with a different right context, leading to very different output distributions in the last state. As the left contexts are identical, the first and second output distributions are almost identical. If we measure the overall model similarity based on the accumulative overall output distribution similarities of all states, these two models may be clustered, leading to a very inaccurate distribution for the last state. Instead, we cluster the first two output distributions while leaving the last one intact.

There are two key issues in creating trainable context-dependent phonetic or subphonetic units:

- We need to enable better parameter sharing and smoothing. As Figure 9.4 illustrates, many phones have similar effects on neighboring phones. If the acoustic realization is indeed identical, we tie them together to improve trainability and efficiency.
- Since the number of triphones in English is very large (over 100,000), there are many new or unseen triphones that are in the test set but not in the training set. It is important to map these unseen triphones into appropriately trained triphones.

As discussed in Chapter 4, a decision tree is a binary tree to classify target objects by asking binary questions in a hierarchical manner. Modeling unseen triphones is particularly important for *vocabulary independence*, since it is difficult to collect a training corpus which covers enough occurrences of every possible subword unit. We need to find models that are accurate, trainable, and especially generalizable. The senonic decision tree classifies Markov states of triphones represented in the training corpus by asking linguistic questions composed of conjunctions, disjunctions, and/or negations of a set of predetermined simple categorical linguistic questions. Examples of these simple categorical questions are: *Is the left-context phone a fricative? Is the right-context phone a front vowel?* The typical question set used in Whisper to generate the senone tree is shown in Table 9.3. So, for each node in the tree, we check whether its left or right phone belongs to one of the categories. As discussed in Chapter 4, we measure the corresponding entropy reduction or likelihood increase for each question and select the question that has the largest entropy decrease to split the node.

Thus, the tree can be automatically constructed by searching, for each node, the question that renders the maximum entropy decrease. Alternatively, complex questions can be formed for each node for improved splitting.

When we grow the tree, it needs to be pruned using cross-validation as discussed in Chapter 4. When the algorithm terminates, the leaf nodes of the tree represent the senones to be used. Figure 9.6 shows an example tree we built to classify the second state of all */k/* triphones seen in a training corpus. After the tree is built, it can be applied to the second state of *any* */k/* triphone, thanks to the generalizability of the binary tree and the general linguistic questions. Figure 9.6 indicates that the second state of the */k/* triphone in *welcome* is mapped to the second senone, no matter whether this triphone occurs in the training corpus or not.

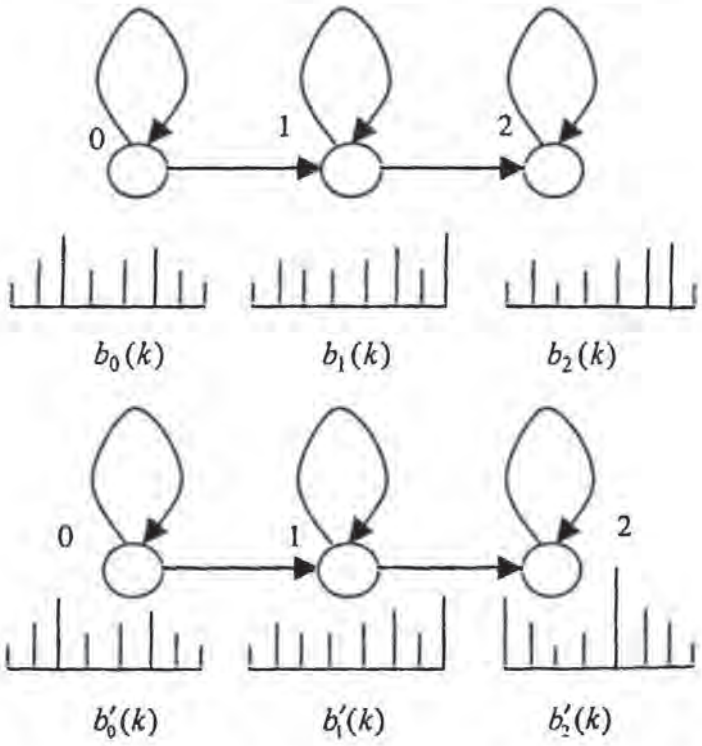


Figure 9.5 State-based vs. model-based clustering. These two models are very similar, as both the first and the second output distributions are almost identical. The key difference is the output distribution of the third state. If we measure the overall model similarity, which is often based on the accumulative output distribution similarities of all states, these two models may be clustered, leading to a very inaccurate distribution for the last state. If we cluster output distributions at state level, we can cluster the first two output distributions while leaving the last ones intact, leading to more accurate representations.

Table 9.3 Some example questions used in building senone trees.

Questions	Phones in Each Question Category
Aspseg	*hh*
Sil	*sil*
Alvstp	*d t*
Dental	*dh th*
Labstp	*b p*
Liquid	*l r*
Lw	*l w*
S/Sh	*s sh*
Sylbic	*er axr*
Velstp	*g k*
Affric	*ch jh*
Lqgl-B	*l r w*
Nasal	*m n ng*
Retro	*r er axr*
Schwa	*ax ix axr*
Velar	*ng g k*
Fric2	*th s sh f*
Fric3	*dh z zh v*
Lqgl	*l r w y*
S/Z/Sh/Zh	*s z sh zh*
Wglide	*uw aw ow w*
Labial	*w m b p v*
Palatl	*y ch jh sh zh*
Yglide	*iy ay ey oy y*
High	*ih ix iy uh uw y*
Lax	*eh ih ix uh ah ax*
Low	*ae aa ao aw ay oy*
Orstp2	*p t k*
Orstp3	*b d g*
Alvelr	*n d t s z*
Diph	*uw aw ay ey iy ow oy*
Fric1	*dh th s sh z zh v f*
Round	*uh ao uw ow oy w axr er*
Frnt-R	*ae eh ih ix iy ey ah ax y aw*
Tense	*iy ey ae uw ow aa ao ay oy aw*
Back-L	*uh ao uw ow aa er axr l r w aw*
Frnt-L	*ae eh ih ix iy ey ah ax y oy ay*
Back-R	*uh ao uw ow aa er axr oy l r w ay*
Orstp1	*b d g p t k ch jh*
Vowel	*ae eh ih ix iy uh ah ax aa ao uw aw ay ey ow oy er axr*
Son	*ae eh ih ix iy ey ah ax oy ay uh ao uw ow aa er axr aw l r w y*
Voiced	*ae eh ih ix iy uh ah ax aa ao uw aw ay ey ow oy l r w y er axr m n ng jh b d dh g v z zh*

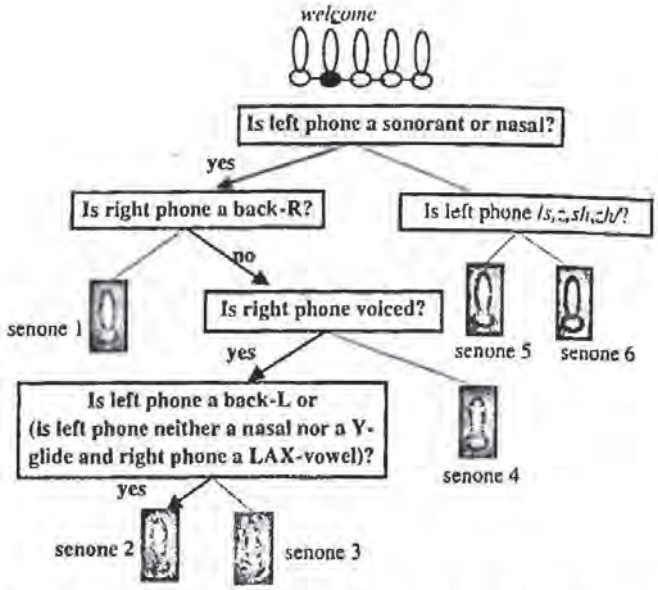


Figure 9.6 A decision tree for classifying the second state of K-triphone HMMs [48].

In practice, senone models significantly reduce the word recognition error rate in comparison with model-based clustered triphone models, as illustrated in Table 9.4. It is the senonic model's significant reduction of the overall system parameters that enables the continuous mixture HMMs to perform well for large-vocabulary speech recognition [56].

Table 9.4 Relative error reductions for different modeling units. The reduction is relative to that of the preceding row.

Units	Relative Error Reductions
Context-independent phone	Baseline
Context-dependent phone	+25%
Clustered triphone	+15%
Senone	+24%

9.4.4. Lexical Baseforms

When appropriate subword units are used, we must have the correct pronunciation for each word so that concatenation of subword units can accurately represent the word to be recognized. The dictionary represents the standard pronunciation used as a starting point for building a workable speech recognition system. We also need to provide alternative pronunciations for words such as *tomato* that may have very different pronunciations. For example, the COMLEX dictionary from LDC has about 90,000 baseforms that cover most words used in many years of *The Wall Street Journal*. The CMU Pronunciation Dictionary, which was optimized for continuous speech recognition, has about 100,000 baseforms.

In continuous speech recognition, we must also use phonologic rules to modify inter-word pronunciations or to have reduced sounds. Assimilation is a typical coarticulation phenomenon—a change in a segment to make it more like a neighboring segment. Typical examples include phrases such as *did you /d ih jh y ah/, set you /s eh ch er/, last year / l ae s ch iy r/, because you've /b iy k ah zh uw v/,* etc. Deletion is also common in continuous speech. For example, */t/* and */d/* are often deleted before a consonant. Thus, in conversational speech, you may find examples like *find him /f ay n ix m/, around this /ix r aw n ih s/,* and *Let me in /l eh m eh n/.*

Dictionaries often don't include proper names. For example, the 20,000 names included in the COMLEX dictionary are a small fraction of 1–2 million names in the USA. To deal with these new words, we often have to derive their pronunciation automatically. These new words have to be added on the fly, either by the user or through an interface from speech-aware applications. Unlike Spanish or Italian, rule-based letter-to-sound (LTS) conversion for English is often impractical, since so many words in English don't follow phonological rules. A trainable LTS converter is attractive, since its performance can be improved by constantly learning from examples so that it can generalize rules for the specific task. Trainable LTS converters can be based on neural networks, HMMs, or the CART described in Chapter 4. In practice, CART-based LTS has a very accurate performance [10, 61, 71, 89].

When CART is used, the basic YES-NO question for LTS conversion looks like: *Is the second right letter 'p'?* or: *Is the first left output phone /ay/?* The question for letters and phones can be on either the left or the right side. The range of question positions should be long enough to cover the most important phonological variations. Empirically, a 10-letter window (5 for left letter context and 5 for right letter context) and 3-phone window context is generally sufficient. A primitive set of questions can include all the singleton questions about each letter or phone identity. If we allow the node to have a complex question—that is, a combination of primitive questions—the depth of the tree can be greatly reduced and performance improved. For example, a complex question: *Is the second left letter 't' and the first left letter 'i' and the first right letter 'n'?* can capture *o* in the common suffix *tion* and convert it to the correct phone. Complex questions can also alleviate possible data-fragmentation problems caused by the greedy nature of the CART algorithm.

Categorical questions can be formed in both the letter and phone domains with our common linguistic knowledge. For example, the most often used set includes the letter or phone clusters for vowels, consonants, nasals, liquids, fricatives, and so on. In growing the decision tree, the context distance also plays a major role in the overall quality. It is very important to weight the entropy reduction according to the distance (either letter or phoneme) to avoid overgeneralization, which forces the tree to look more carefully at the *nearby* context than at the *far-away* context. Each leaf of the tree has a probability distribution for letter-to-phoneme mapping.

There are a number of ways to improve the effectiveness of the decision tree. First, pruning controls the tree's depth. For example, certain criteria have to be met for a node to be split. Typically splitting requires a minimum number of counts and a minimum entropy reduction. Second, the distribution at the leaves can be smoothed. For example, a leaf distri-

bution can be interpolated with the distributions of its ancestor nodes using deleted interpolation. Finally, we can partition the training data and build multiple trees with different prediction capabilities. These trees accommodate different phonological rules with different language origins.

When the decision tree is used to derive the phonetic pronunciation, the phonetic conversion error is about 8% for the *Wall Street Journal* newspaper text corpora [61]. These errors can be broadly classified into two categories. The first includes errors of proper nouns and foreign words. For example, *Pacino* can be mistakenly converted to */p ax s iy n ow/* instead of */p ax ch iy n ow/*. The second category includes generalization errors. For example, *shier* may be converted to */sh ih r/* instead of the correct pronunciation */sh ay r/* if the word *cashier /k ae sh ih r/* appears in the training data. The top three phone confusion pairs are */ix/ax/*, */dx/t/*, and */ae/ax/*. The most confusing pair is */ix/ax/*. This is not surprising, because */ix/ax/* is among the most inconsistent transcriptions in most of the published dictionaries. There is no consensus for */ix/ax/* transcription among phoneticians.

Although automatic LTS conversion has a reasonable accuracy, it is hardly practical if you don't use an exception dictionary. This is especially true for proper nouns. In practice, you can often ask the person who knows how to pronounce the word to either speak or write down the correct phonetic pronunciation, updating the exception dictionary if the correct one disagrees with what the LTS generates. When acoustic examples are available, you can use the decision tree to generate multiple results and use these results as a language model to perform phone recognition on the acoustic examples. The best overall acoustic and LTS probability can be used as the most likely candidate in the exception dictionary. Since there may be many ways to pronounce a word, you can keep multiple pronunciations in the dictionary with a probability for each possible one. If the pronunciation probability is inaccurate, an increase in multiple pronunciations essentially increases the size and confusion of the vocabulary, leading to increased speech recognition error rate.

Even if you have accurate phonetic baseforms, pronunciations in spontaneous speech differ significantly from the standard baseform. Analysis of manual phonetic transcription of conversational speech reveals a large number (> 20%) of cases of genuine ambiguity: instances where human labelers disagree on the identity of the surface form [95]. For example, the word *because* has more than 15 different pronunciation variations, such as */b iy k ah z/, /b ix k ah z/, /k ah z/, /k ax z/, /b ix k ax z/, /b ax k ah z/, /b ih k ah z/, /k s/, /k ix z/, /k ih z/, /b iy k ah s/, /b iy k ah/, /b iy k ah zh/, /ax z/*, etc., in the context of conversational speech [39]. To characterize the acoustic evidence in the context of this ambiguity, you can partly resolve the ambiguity by deriving a suitable phonetic baseform from speech data [29, 95, 97]. This is because the widespread variation can be due either to a lexical fact (such as that the word *because* can be *'cause* in informal speech) or to the dialect differences. African American vernacular English has many vowels different from general American English.

To incorporate widespread pronunciations, we can use a probabilistic finite state machine to model each word's pronunciation variations, as shown in Figure 9.7. The probability with each arc indicates how likely that path is to be taken, with all the arcs that leave a node summing to 1. As with HMMs, these weights can be estimated from a real corpus for

improved speech recognition [20, 85, 102, 103, 110]. In practice, the relative error reduction of using probabilistic finite state machines is very modest (5–10%).

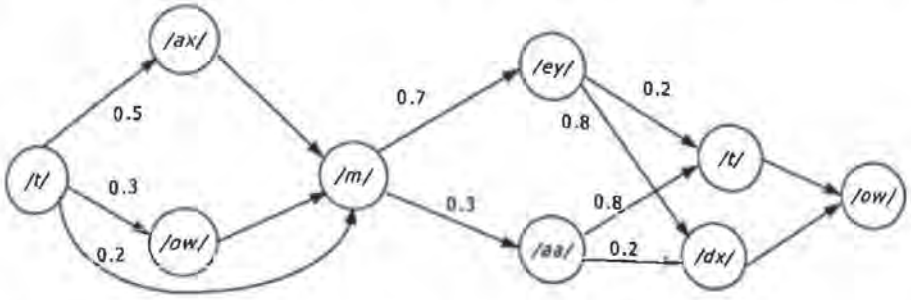


Figure 9.7 A possible pronunciation network for word *tomato*. The vowel */ey/* is more likely to flap, thereby having a higher transition probability into */dx/*.

9.5. ACOUSTIC MODELING—SCORING ACOUSTIC FEATURES

After feature extraction, we have a sequence of feature vectors, **X**, such as the MFCC vector, as our input data. We need to estimate the probability of these acoustic features, given the word or phonetic model, **W**, so that we can recognize the input data for the correct word. This probability is referred to as acoustic probability, $P(\mathbf{X} \mid \mathbf{W})$. In this section we focus our discussion on the HMM. As discussed in Chapter 8, it is the most successful method for acoustic modeling. Other emerging techniques are discussed in Section 9.8.

9.5.1. Choice of HMM Output Distributions

As discussed in Chapter 8, you can use discrete, continuous, or semicontinuous HMMs. When the amount of training data is sufficient, parameter tying becomes unnecessary. A continuous model with a large number of mixtures offers the best recognition accuracy, although its computational complexity also increases linearly with the number of mixtures. On the other hand, the discrete model is computationally efficient, but has the worst performance among the three models. The semicontinuous model provides a viable alternative between system robustness and trainability.

When either the discrete or the semicontinuous HMM is employed, it is helpful to use multiple codebooks for a number of features for significantly improved performance. Each codebook then represents a set of different speech parameters. One way to combine these multiple output observations is to assume that they are independent, computing the output probability as the product of the output probabilities of each codebook. For example, the semicontinuous HMM output probability of multiple codebooks can be computed as the product of each codebook:

$$b_i(\mathbf{x}) = \prod_m \sum_{k=1}^{L^m} f^m(\mathbf{x}^m \mid o_k^m) b_i^m(o_k^m) \tag{9.12}$$

where superscript m denotes the codebook-m related parameters. Each codebook consists of L^m-mixture continuous density functions.

Following our discussion in Chapter 8, the re-estimation algorithm for the multiple-codebook-based HMM could be extended. Since multiplication of the output probability density of each codebook leads to several independent terms in the Q-function, for codebook m, $\zeta_t(j,k^m)$ can be modified as follows:

$$\zeta_t(j,k^m) = \frac{\sum_i \alpha_{t-1}(i) a_{ij} b_j^m(k^m) f^m(\mathbf{x}_t \mid v_k^m) \prod_{n \neq m} \sum_k b_j^n(k^n) f^n(\mathbf{x}_t \mid v_k^n) \beta_t(j)}{\sum_k \alpha_T^m(k)} \tag{9.13}$$

Other intermediate probabilities can also be computed in a manner similar to what we discussed in Chapter 8.

Multiple codebooks can dramatically increase the representation power of the VQ codebook and can substantially improve speech recognition accuracy. You can typically build a codebook for $\mathbf{c}_k$, $\Delta\mathbf{c}_k$, and $\Delta\Delta\mathbf{c}_k$, respectively. As energy has a very different dynamic range, you can further improve the performance by building a separate codebook for $c_k[0]$, $\Delta c_k[0]$, and $\Delta\Delta c_k[0]$. In comparison to building a single codebook for $\mathbf{x}_k$ as illustrated in Eq. (9.6), the multiple-codebook system can reduce the error rate by more than 10%.

In practice, the most important parameter for the output probability distribution is the number of mixtures or the size of the codebooks. When there are sufficient training data, relative error reductions with respect to the discrete HMM are those shown in Figure 9.8.

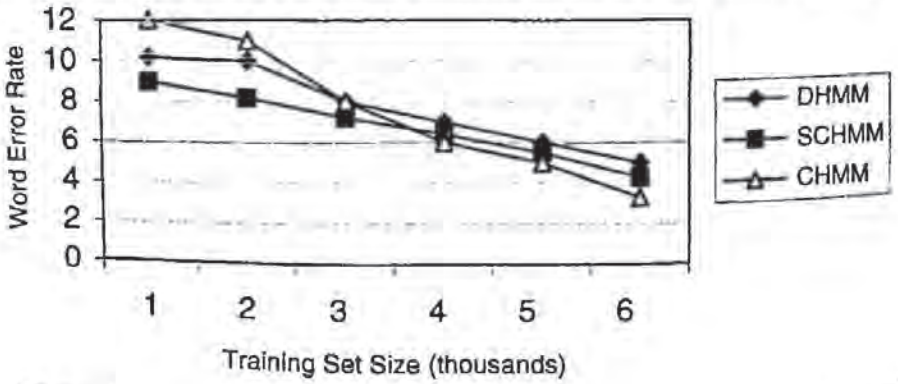


Figure 9.8 Continuous speaker-independent word recognition error rates of the discrete HMM (DHMM), SCHMM, and the continuous HMM (CHMM) with respect to the training set sizes (thousands of training sentences). Both the DHMM and SCHMM have multiple codebooks. The CHMM has 20 mixture diagonal Gaussian density functions.

As you can see from Figure 9.8, the SCHMM offers improved accuracy in comparison with the discrete HMM or the continuous HMM when the amount of training data is limited. When we increase the training data size, the continuous mixture density HMM starts to outperform both the discrete and the semicontinuous HMM, since the need to share model parameters becomes less critical.

Performance is also a function of the number of mixtures. With a small number of mixtures, the continuous HMM lacks the modeling power and it actually performs worse than the discrete HMM across the board. Only after we dramatically increase the number of mixtures does the continuous HMM start to offer improved recognition accuracy. The SCHMM can typically reduce the discrete HMM error rate by 10–15% across the board. The continuous HMM with 20 diagonal Gaussian density functions performed worse than either the discrete or the SCHMM when the size of training data was small. It outperformed either the discrete HMM or the SCHMM when sufficient amounts of training data became available. When the amount of training data is sufficiently large, it can reduce the error rate of the semicontinuous HMM by 15–20%.

9.5.2. Isolated vs. Continuous Speech Training

If we build a word HMM for each word in the vocabulary for isolated speech recognition, the training or recognition can be implemented directly, using the basic algorithms introduced in Chapter 8. To estimate model parameters, examples of each word in the vocabulary are collected. The model parameters are estimated from all these examples using the forward-backward algorithm and the reestimation formula. It is not necessary to have precise end-point detection, because the silence model automatically determines the boundary if we concatenate silence models with the word model in both ends.

If subword units,[3] such as phonetic models, are used, we need to share them across different words for large-vocabulary speech recognition. These subword units are concatenated to form a word model, possibly adding silence models at the beginning and end, as illustrated in Figure 9.9.

To concatenate subword units to form a word model, you can have a null transition from the final state of the previous subword HMM to the initial state of the next subword HMM, as indicated by the dotted line in Figure 9.9. As described in Chapter 8, you can estimate the parameters of the concatenated HMM accordingly. Please notice that the added null transition arc should satisfy the probability constraint with the transition probability of each phonetic HMM. The self-loop transition probability of the last state in each individual HMM has the topology illustrated in Figure 9.9. If we estimate these parameters with the concatenated model, the null arc transition probability, a_{ij}^{ε}, should satisfy the constraint $\sum_j (a_{ij} + a_{ij}^{\varepsilon}) = 1$ such that the self-loop transition probability of the last state is no longer equal to 1. For interword concatenation or concatenation involving multiple pronunciations, you can use multiple null arcs to concatenate individual models together.

[3] We have a detailed discussion on word models vs. subword models in Section 9.4.1.

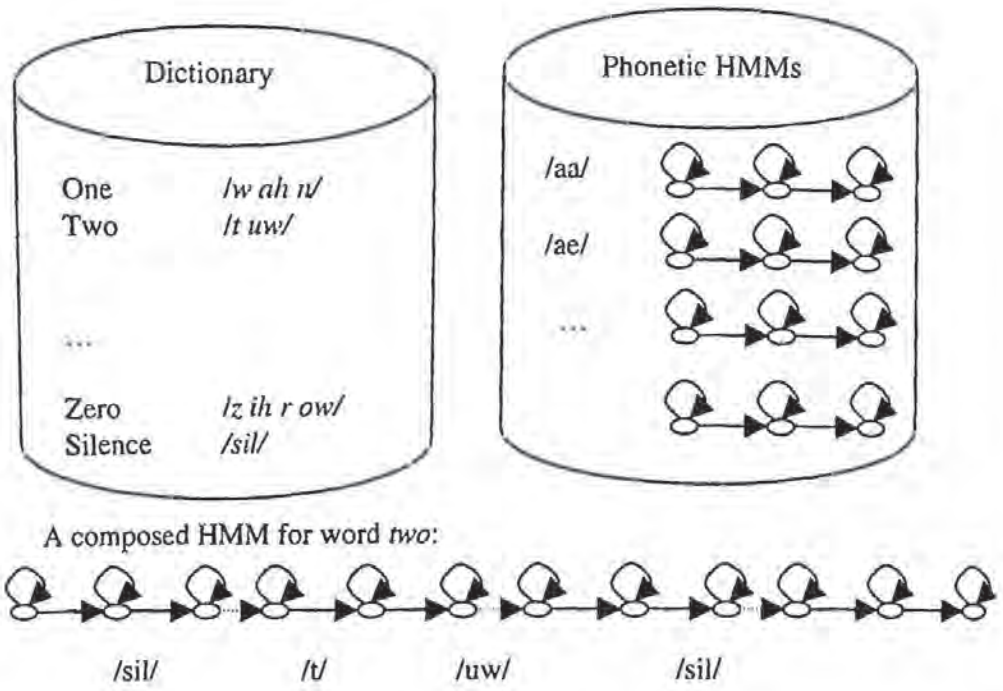


Figure 9.9 The construction of an isolated word model by concatenating multiple phonetic models based on the pronunciation dictionary.

In the example given in Figure 9.9, we have ten English digits in the vocabulary. We build an HMM for each English phone. The dictionary provides information on each word's pronunciation. We have a special word, *Silence*, that maps to a */sil/* HMM that has the same topology as the standard phonetic HMM. For each word in the vocabulary we first derive the phonetic sequence for each word from the dictionary. We link these phonetic models together to form a word HMM for each word in the vocabulary. The link between two phonetic models is shown in the figure as the dotted arrow.

For example, for word *two*, we create a word model based on the beginning silence */sil/*, phone */t/*, phone */uw/*, and ending silence */sil/*. The concatenated word model is then treated in the same manner as a standard large composite HMM. We use the standard forward-backward algorithm to estimate the parameters of the composite HMM from multiple sample utterances of the word *two*. After several iterations, we automatically get the HMM parameters for */sil/*, */t/*, and */uw/*. Since a phone can be shared across different words, the phonetic parameters may be estimated from acoustic data in different words.

The ability to automatically align each individual HMM to the corresponding unsegmented speech observation sequence is one of the most powerful features in the forward-backward algorithm. When the HMM concatenation method is used for continuous speech, you need to compose multiple words to form a sentence HMM based on the transcription of the utterance. In the same manner, the forward-backward algorithm absorbs a range of pos-

sible word boundary information of models automatically. There is no need to have a precise segmentation of the continuous speech.

In general, to estimate the parameters of the HMM, each word is instantiated with its concatenated word model (which may be a concatenation of subword models). The words in the sentence are concatenated with optional silence models between them. If there is a need to modify interword pronunciations due to interword pronunciation change, such as *want you*, you can add a different optional phonetic sequence for *t-y* in the concatenated sentence HMM.

In the digit recognition example, if we have a continuous training utterance *one three*, we compose a sentence HMM, as shown in Figure 9.10, where we have an optional silence HMM between the words *one* and *three*, linked with a null transition from the last state of the word model *one* to the first state of the word model *three*. There is also a direct null arc connection between the models *one* and *three* because a silence may not exist in the training example. These optional connections ensure that all the possible acoustic realizations of the natural continuous speech are considered, so that the forward-backward algorithm can automatically discover the correct path and accurately estimate the corresponding HMM from the given speech observation.

In general, the concatenated sentence HMM can be trained using the forward-backward algorithm with the corresponding observation sequence. Since the entire sentence HMM is trained on the entire observation sequence for the corresponding sentence, most possible word boundaries are inherently considered. Parameters of each model are based on those state-to-speech alignments. It does not matter where the word boundaries are. Such a training method allows complete freedom to align the sentence model against the observation, and no explicit effort is needed to find word boundaries.

In speech decoding, a word may begin and end anywhere within a given speech signal. As word boundaries cannot be detected accurately, all possible beginning and end points have to be accounted for. This converts a linear search (as for isolated word recognition) to a tree search, and a polynomial recognition algorithm to an exponential one. How to design an efficient decoder is discussed in Chapters 12 and 13.

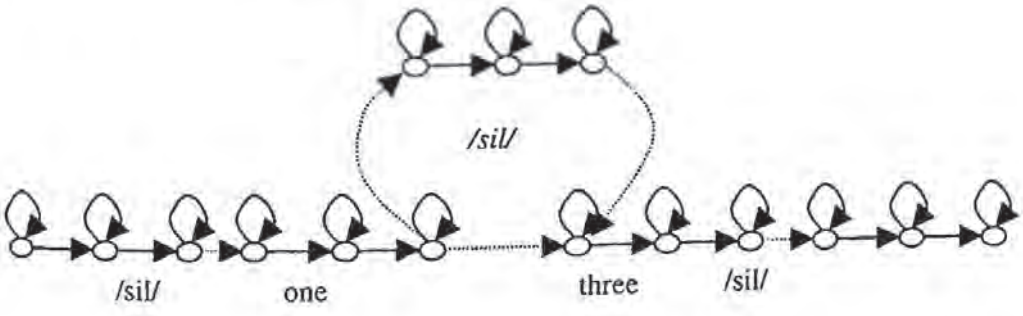


Figure 9.10 A composite sentence HMM. Each word can be a word HMM or a composite phonetic word HMM, as illustrated in Figure 9.9.

9.6. ADAPTIVE TECHNIQUES—MINIMIZING MISMATCHES

As Figure 1.2 illustrated, it is important to adapt both acoustic models and language models for new situations. A decent model can accommodate a wide range of variabilities. However, the mismatch between the model and operating conditions always exists. One of the most important factors in making a speech system usable is to minimize the possible mismatch dynamically with a small amount of *calibration data*. Adaptive techniques can be used to modify system parameters to better match variations in microphone, transmission channel, environment noise, speaker, style, and application contexts. As a concrete example, speaker-dependent systems can provide a significant word error-rate reduction in comparison to speaker-independent systems if a large amount of speaker-dependent training data exists [50]. Speaker-adaptive techniques can bridge the gap between these two configurations with a small fraction of the speaker-specific training data needed to build a full speaker-dependent system. These techniques can also be used incrementally as more speech is available from a particular speaker. When speaker-adaptive models are built, you can have not only improved accuracy but also improved speed and potentially reduced model parameter sizes because of accurate representations, which is particularly appealing for practical speech recognition.

There are a number of ways to use adaptive techniques to minimize mismatches. You can have a nonintrusive adaptation process that works in the background all the time. This is typically unsupervised, using only the outcome of the recognizer (with a high confidence score, as discussed in Section 9.7) to guide the model adaptation. This approach can continuously modify the model parameters so that any nonstationary mismatches can be eliminated. As discussed in Chapter 13, systems that are required to transcribe speech in a non-real-time fashion may use multiple recognition passes. You can use unsupervised adaptation on the test data to improve the models after each pass to improve performance for a subsequent recognition pass.

Since the use of recognition results may be imperfect, there is a possibility of divergence if the recognition error rate is high. If the error rate is low, the adaptation results may still not be as good as supervised adaptation in which the correct transcription is provided for the user to read, a process referred to as the *enrollment process*. In this process you can check a wide range of parameters as follows:

- Check the background noise by asking the user not to speak.
- Adjust the microphone gain by asking the user to speak normally.
- Adapt the acoustic parameters by asking the user to read several sentences.
- Change the decoder parameters for the best speed with no loss of accuracy.
- Compose dynamically new enrollment sentences based on the user-specific error patterns.

The challenge for model adaptation is that we can use only a small amount of observable data to modify model parameters. This constraint requires different modeling strategies

from the ones we discussed in building the baseline system, as the amount of training data is generally sufficient for offline training. In this section we focus on a number of adaptive techniques that can be applied to compensate either speaker or environment variations. Most of these techniques are model-based, since the acoustic model parameters rather than the acoustic feature vectors are adapted. We use speaker-adaptation examples to illustrate how these techniques can be used to improve system performance. We can generalize to environment adaptation by using environment-specific adaptation data and a noise-compensation model, which we discuss in Chapter 10. In a similar manner, we can modify the language model as discussed in Chapter 11.

9.6.1. Maximum a Posteriori (MAP)

Maximum a posteriori (MAP) estimation, as discussed in Chapter 4, can effectively deal with data-sparse problems, as we can take advantage of prior information about existing models. We can adjust the parameters of pretrained models in such a way that limited new training data would modify the model parameters guided by the prior knowledge to compensate for the adverse effect of a mismatch [35]. The prior density prevents large deviations of the parameters unless the new training data provide strong evidence.

More specifically, we assume that an HMM is characterized by a parameter vector Φ that is a random vector, and that prior knowledge about the random vector is available and characterized by a prior probability density function $p(\Phi)$, whose parameters are to be determined experimentally.

With the observation data $\mathbf{X}$, the MAP estimate is expressed as follows:

$$\hat{\Phi} = \arg\max_{\Phi}[p(\Phi \mid \mathbf{X})] = \arg\max_{\Phi}[p(\mathbf{X} \mid \Phi)p(\Phi)] \tag{9.14}$$

If we have no prior information, $p(\Phi)$ is the uniform distribution, and the MAP estimate becomes identical to the ML estimate. We can use the EM algorithm as the ML to estimate the parameters of HMMs. The corresponding Q-function can be defined as:

$$Q_{MAP}(\Phi, \hat{\Phi}) = \log p(\hat{\Phi}) + Q(\Phi, \hat{\Phi}) \tag{9.15}$$

The EM algorithm for the ML criterion can be applied here directly. The actual expression depends on the assumptions made about the prior density. For the widely used continuous Gaussian mixture HMM, there is no joint conjugate prior density. We can assume different components of the HMM to be mutually independent, so that the optimization can be split into different subproblems involving only a single component of the parameter set. For example, the prior density function for the mixture Gaussian can be as follows:

$$p_{b_i}(\mathbf{c}_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = p_{c_i}(\mathbf{c}_i)\prod_k p_{b_{ik}}(\boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}) \tag{9.16}$$

where $p_{c_i}(\mathbf{c}_i)$ is a Dirichlet prior density for the mixing coefficient vector of all mixture components in the Markov state i, and $p_{b_{ik}}(\boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik})$ denotes the prior density for parameters

of the kth Gaussian component in the state i. The Dirichlet prior density $p_{c_i}(\mathbf{c}_i)$ is characterized by a vector v_i of positive hyperparameters such that:

$$p_{c_i}(\mathbf{c}_i) \propto \prod_k c_{ik}^{v_k - 1} \tag{9.17}$$

For full covariance D-dimensional Gaussian densities, the prior density can be a normal-Wishart density parameterized by two values $\eta > D-1, \tau > 0$, the vector μ_{mw}, and the symmetric positive definite matrix $\mathbf{S}$ as follows:

$$p_{b_{ik}}(\mu_{ik}, \Sigma_{ik}) \propto$$
$$\sqrt{\det(\Sigma_{ik})^{D-\eta}} \exp(-\frac{\tau}{2}(\mu_{ik} - \mu_{mw})\Sigma_{ik}^{-1}(\mu_{ik} - \mu_{mw})' - \frac{1}{2} tr(\mathbf{S}\Sigma_{ik}^{-1})) \tag{9.18}$$

We can apply the same procedure as the MLE Baum-Welch reestimation algorithm. For example, with the Q-function defined in Eq. (9.15), we can apply the Lagrange method to derive the mixture coefficients as follows:

$$\begin{cases} \dfrac{\partial}{\partial \hat{c}_{ik}}(\log p_{c_i}(\hat{\mathbf{c}}_i) + \sum_k \sum_t \xi_t(i,k) \log \hat{c}_{ik}) + \lambda = 0, \forall k \\ \sum_k \hat{c}_{ik} = 1 \end{cases} \tag{9.19}$$

Based on Eqs. (9.17) and (9.19), the solution is:

$$\hat{c}_{ik} = \frac{v_{ik} - 1 + \sum_t \xi_t(i,k)}{\sum_l (v_{il} - 1 + \sum_t \xi_t(i,l))} \tag{9.20}$$

A comparison between Eq. (9.20) and the ML estimate Eq. (8.58) shows that the MAP estimate is a weighted average between the mode of the prior density and the ML estimate with proportions given by $v_{ijk} - 1$ and $\sum \xi_t(i,k)$, respectively.

We can optimize Eq. (9.15) with respect to mean and covariance parameters in a similar fashion. For example, the solution of these estimates is:

$$\hat{\mu}_{ik} = \frac{\tau_{ik}\mu_{mw_{ik}} + \sum_{t=1}^{T} \zeta_t(i,k)\mathbf{x}_t}{\tau_{ik} + \sum_{t=1}^{T} \zeta_t(i,k)} \tag{9.21}$$

$$\hat{\Sigma}_{ik} = \frac{\mathbf{S}_{ik} + \tau_{ik}(\hat{\mu}_{ik} - \mu_{mw_{ik}})(\hat{\mu}_{ik} - \mu_{mw_{ik}})' + \sum_{t=1}^{T} \zeta_t(i,k)(\mathbf{x} - \hat{\mu}_{ik})(\mathbf{x} - \hat{\mu}_{ik})'}{\eta_{ik} - D + \sum_{t=1}^{T} \zeta_t(i,k)} \tag{9.22}$$

where τ_{ik} is the parameter in the normal-gamma density for the corresponding state i.

Thus, the reestimation formula for the Gaussian mean is a weighted sum of the prior mean with the ML mean estimate $\sum_{t=1}^{T}\zeta_t(i,k)\mathbf{x}_t / \sum_{t=1}^{T}\zeta_t(i,k)$. τ_{ik} is a balancing factor between prior mean and the ML mean estimate. When τ_{ik} is large, the value of the prior knowledge is small and the value of the mean μ_{imix} is assumed to have high certainty, leading to the dominance of the final estimate. When the amount of adaptation data increases, the MAP estimate approaches the ML estimate, as the adaptation data overwrite any important prior that may influence the final estimate. Similarly, the covariance estimation formula has the same interpretation of the balance between the prior and new data.

One major limitation of the MAP-based approach is that it requires an accurate initial guess for the prior $p(\Phi)$, which is often difficult to obtain. We can use the already trained initial models that embody some characteristics of the original training conditions. A typical way to generate an initial Gaussian prior is to cluster the initial training data based on speaker or environment similarity measures. We can derive a set of models based on the partition, which can be seen as a set of observations drawn from a distribution having $p(\Phi)$. We can, thus, estimate the prior based on the sample moments to derive the corresponding prior parameters.

Another major limitation is that the MAP-based approach is a local approach to updating the model parameters. Namely, only model parameters that are observed in the adaptation data can be modified from the prior value. When the system has a large number of free parameters, the adaptation can be very slow. Thus in practice we need to find correlations between the model parameters, so that the unobserved or poorly adapted parameters can be altered [3, 22]. Another possibility is to impose structural information so the model parameters can be shared for improved adaptation speed [96].

The MAP training can be iterative, too, which requires an initial estimate of model parameters. A careful initialization for the Gaussian densities is also very important. Unlike the discrete distributions, there is no such a thing as a uniform density for a total lack of information about the value of the parameters. We need to use the same initialization procedure as discussed in Chapter 8.

For speaker-adaptive speech recognition, it has been experimentally found that τ_{ik} can be a fixed constant value for all the Gaussian components across all the dimensions. Thus the MAP HMM can be regarded as an interpolated model between the speaker-independent and speaker-dependent HMM. Both are derived from the standard ML forward-backward algorithm. Experimental performance of MAP training is discussed in Section 9.6.3.

9.6.2. Maximum Likelihood Linear Regression (MLLR)

When the continuous HMM is used for acoustic modeling, the most important parameter set to adapt is the output Gaussian density parameters, i.e., the mean vector and the covariance matrix. We can use a set of linear regression transformation functions to map both means

and covariances in order to maximize the likelihood of the adaptation data [68]. The maximum likelihood linear regression (MLLR) mapping is consistent with the underlying criterion for building the HMM while keeping the number of free parameters under control. Since the transformation parameters can be estimated from a relatively small amount of adaptation data, it is very effective for rapid adaptation. MLLR has been widely used to obtain adapted models for either a new speaker or a new environment condition.

More specifically, in the mixture Gaussian density functions, the kth mean vector $\boldsymbol{\mu}_{ik}$ for each state i can be transformed using following equation:

$$\tilde{\boldsymbol{\mu}}_{ik} = \mathbf{A}_c \boldsymbol{\mu}_{ik} + \mathbf{b}_c \tag{9.23}$$

where $\mathbf{A}_c$ is a regression matrix and $\mathbf{b}_c$ is an additive bias vector associated with some broad class c, which can be either a broad phone class or a set of tied Markov states. The goal of Eq. (9.23) is to map the mean vector into a new space such that the mismatch can be eliminated. Because the amount of adaptation data is small, we need to make sure the number of broad classes c is small so we have only a small number of free parameters to estimate. Equation (9.23) can be simplified into:

$$\tilde{\boldsymbol{\mu}}_{ik} = \mathbf{W}_c \boldsymbol{\mu}_{ik} \tag{9.24}$$

where $\boldsymbol{\mu}_{ik}$ is extended as $[1, \boldsymbol{\mu}_{ik}^t]^t$ and $\mathbf{W}_c$ is the extended transform, $[\mathbf{b}_c, \mathbf{A}_c]$.

This mapping approach is based on the assumption that $\mathbf{W}_c$ can be tied for a wide range of broad phonetic classes so that the overall number of free parameters is significantly less than the number of mean vectors. Therefore, the same transformation can be used for several distributions if they represent similar acoustic characteristics.

To estimate these transformation parameters in the MLE framework, we can use the same Q-function we discussed in Chapter 8. We need to optimize only

$$\sum_i \sum_{k=1}^{M} Q_{b_i}(\Phi, \hat{\mathbf{b}}_{ik}) \tag{9.25}$$

with respect to $\mathbf{W}_c$. Maximization of $Q_{b_i}(\Phi, \hat{\mathbf{b}}_{ik})$ with respect to $\mathbf{W}_c$ can be achieved by computing the partial derivatives. For the Gaussian mixture density function, the partial derivative with respect to $\mathbf{W}_c$ is:

$$\frac{\partial \hat{b}_{ik}(\mathbf{x})}{\partial \mathbf{W}_c} = N(\mathbf{x}, \tilde{\boldsymbol{\mu}}_{ik}, \hat{\boldsymbol{\Sigma}}_{ik}) \boldsymbol{\Sigma}_{ik}^{-1} (\mathbf{x} - \mathbf{W}_c \boldsymbol{\mu}_{ik}) \boldsymbol{\mu}_{ik}^t \tag{9.26}$$

Let us denote the set of Gaussian components forming the broad transformation classes as C; we use $b_{ik} \in C$ to denote that the k^{th} Gaussian density in state i belongs to the

class. We can expand the Q-function with the partial derivatives and set it to zero, leading to the following equation:

$$\sum_{t=1}^{T}\sum_{b_k \in C}\zeta_t(i,k)\Sigma_{ik}^{-1}\mathbf{x}_t\mu_{ik}' = \sum_{t=1}^{T}\sum_{b_k \in C}\zeta_t(i,k)\Sigma_{ik}^{-1}\mathbf{W}_c\mu_{ik}\mu_{ik}' \tag{9.27}$$

We can rewrite Eq. (9.27) as:

$$\mathbf{Z} = \sum_{b_k \in C}\mathbf{V}_{ik}\mathbf{W}_c\mathbf{D}_{ik} \tag{9.28}$$

where

$$\mathbf{Z} = \sum_{t=1}^{T}\sum_{b_k \in C}\zeta_t(i,k)\Sigma_{ik}^{-1}\mathbf{x}_t\mu_{ik}', \tag{9.29}$$

$$\mathbf{V}_{ik} = \sum_{t=1}^{T}\zeta_t(i,k)\Sigma_{ik}^{-1}, \tag{9.30}$$

and

$$\mathbf{D}_{ik} = \mu_{ik}\mu_{ik}'. \tag{9.31}$$

Estimating $\mathbf{W}_c$ for Eq. (9.28) is computationally expensive, as it requires solving simultaneous equations. Nevertheless, if we assume that the covariance matrix is diagonal, we can have a closed-form solution that is computationally efficient. Thus, we can define

$$\mathbf{G}_q = \sum_{b_{qk} \in C} v_{qq}\mathbf{D}_{ik} \tag{9.32}$$

where v_{qq} denotes the q^{th} diagonal element of matrix $\mathbf{V}_{ik}$. The transformation matrix can be computed row by row. So for the q^{th} row of the transformation matrix $\mathbf{W}_q$, we can derive it from the q^{th} row of $\mathbf{Z}_q$ [defined in Eq. (9.29)] as follows:

$$\mathbf{W}_q = \mathbf{Z}_q\mathbf{G}_q^{-1} \tag{9.33}$$

Since $\mathbf{G}_q$ may be a singular matrix, we need to make sure we have enough training data for the broad class. Thus, if the amount of training data is limited, we must tie a number of transformation classes together.

We can run several iterations to maximize the likelihood for the given adaptation data. At each iteration, transformation matrices can be initialized to identity transformations. We can iteratively repeat the process to update the means until convergence is achieved. We can

also incrementally adapt the mean vectors after each observation sequence or set of observation sequences while the required statistics are accumulated over time. Under the assumption that the alignment of each observation sequence against the model is reasonably accurate, we can accumulate these estimated counts over time and use them incrementally. In order to deal with the tradeoff between specificity and robust estimation, we can dynamically generate regression classes according to the senone tree. Thus, we can incrementally increase the number of regression classes when more and more data become available.

MLLR adaptation can be generalized to include the variances with the ML framework, although the additional gain after mean transformation is often less significant (less than relative 2% error reduction). When the user donates about 15 sentences for enrollment training, Table 9.5 illustrates how the MLLR adaptation technique can be used to further reduce the word recognition error rate for a typical dictation application. Here, there is only one context-independent phonetic class for all the context-dependent Gaussian densities. As we can see, most of the error reduction came from adapting the mean vectors.

We can further extend MLLR to *speaker-adaptive training* (SAT) [6, 74]. In conventional speaker-independent training, we simply use data from different speakers to build a speaker-independent model. An inherent difficulty in this approach is that spectral variations of different speakers give the speaker-independent acoustic model higher variance than the corresponding speaker-dependent model. We can include MLLR transformation in the process of training to derive the MLLR parameters for each individual speaker. Thus the training data are transformed to maximize the likelihood for the overall speaker-independent model. This process can be run iteratively to reduce mismatches of different speakers. By explicitly accounting for the interspeaker variations during training and decoding, SAT reduces the error rate by an additional 5–10%.

Table 9.5 Relative error reductions with MLLR methods. The reduction is relative to that of the preceding row.

Models	Relative Error Reduction
CHMM	Baseline
MLLR on mean only	+12%
MLLR on mean and variance	+2%
MLLR SAT	+8%

9.6.3. MLLR and MAP Comparison

The MLLR method can be combined with MAP. This guarantees that with the increased amount of training data, we can have not only a set of compact MLLR transformation functions for rapid adaptation, but also directly modified model parameters that converge to the ML estimates. We can use MAP to adapt the model parameters and then add MLLR to transform these adapted models. It is also possible to incorporate the MAP principle directly into MLLR [18, 19].

As an example, the result of a 60,000-word dictation application using various adaptation methods is shown in Figure 9.11.[4] The speaker-dependent model used 1000 utterances. Also included as a reference is the speaker-independent result, which is used as the starting point for adaptive training. When the speaker-independent model is adapted with about 200 utterances, the speaker-adaptive model has already outperformed both speaker-independent and speaker-dependent systems. The results clearly demonstrate that we have insufficient training data for speaker-dependent speech recognition, as MAP-based outperform ML-based models. This also illustrates that we can make effective use of speaker-independent data for speaker-dependent speech recognition. Also, notice that the MLLR method has a faster adaptation rate than the MAP method. The MLLR method has context-independent phonetic classes. So, when the amount of adaptation data is limited, the MLLR method offers better overall performance.

However, the MAP becomes more accurate when the amount of adaptation data increases to 600 per speaker. This is because we can modify all the model parameters with the MAP training, and the MLLR transformation can never have the same degrees of freedom as the MAP method. When the MLLR is combined with MAP, we can have not only rapid adaptation but also superior performance over either the MLLR or MAP method across a wide range of adaptation data points. There are a number of different ways to combine both MLLR and MAP for improved performance [4, 98].

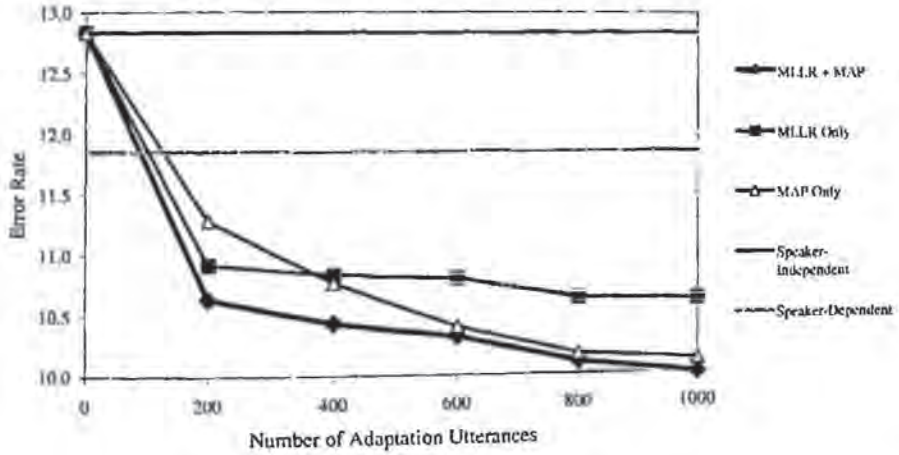


Figure 9.11 Comparison of Whisper with MLLR, MAP, and combined MLLR and MAP. The error rate is shown for a different amount of adaptation data. The speaker-dependent and speaker-independent models are also included. The speaker-dependent model was trained with 1000 sentences.

[4] In practice, if a large well-trained, speaker-independent model is used, the baseline performance may be very good, and hence, the relative error reduction from speaker adaptation may be smaller than for smaller and simpler models.

9.6.4. Clustered Models

Both MAP and MLLR techniques are based on using an appropriate initial model for adaptive modeling. How accurate we make the initial model directly affects the overall performance. An effective way to minimize the mismatch is, thus, to cluster similar speakers and environments in the training data, building a set of models for each cluster that has minimal mismatch for different conditions. When we have enough training data, and enough coverage for a wide range of conditions, this approach ensures significantly improved robustness.

For example, we often need a set of clustered models for different telephone channels, including different cellular phone standards. We also need to build gender-dependent models or speaker-clustered models for improved performance. In fact, when we construct speaker-clustered models, we can apply MLLR transformations or neural networks to minimize speaker variations such that different speakers can be mapped to the same golden speaker that is the representative of the cluster.

Speaker clusters can be created based on the information of each speaker-dependent HMM. The clustering procedure is similar to the decision-tree procedure discussed in Section 9.4.3. Using clustered models increases the amount of computational complexity. It also fragments the training data. Clustering is often needed to combine other smoothing techniques, such as deleted interpolation or MLLR transformation, in order to create clustered models from the pooled model. We can also represent a speaker as a weighted sum of individual speaker cluster models with the cluster adaptive training [33] or eigenvoice techniques [64].

When we select an appropriate model, we can compute the likelihood of the test speech against all the models and select the model that has the highest likelihood. Alternatively, we can compute likelihoods as part of the decoding process and prune away less promising models dynamically without significantly increasing the computational load. When multiple models are plausible, we can compute the weighted sum of the clustered models with pretrained mixing coefficients for different clusters, much as we train the deleted interpolation weights.

Traditionally speaker clustering is performed across different speakers without considering phonetic similarities across different speakers. In fact, clustered speaker groups may have very different degrees of variations for different phonetic classes. You can further generalize speaker clustering to the subword or subphonetic level [62]. With multiple instances derived from clustering for each subword or subphonetic unit, you can model speaker variation explicitly across different subword or subphonetic models.

In practice, gender-dependent models can reduce the word recognition error by 10%. More refined speaker-clustered models can further reduce the error rate, but not as much as the gain from gender-dependent models, unless we have a large number of clustered speakers. If the new user happens to be similar to one of these speaker clusters, we can approach speaker-dependent speech recognition without enrollment. For environment-dependent models, clustering is more critical. The challenge is to anticipate the kind of environment or channel distortions the system will have to deal with. Since this is often unpredictable, we

need to use adaptive techniques such as MAP and MLLR to minimize the mismatch. We discuss this in more detail in Chapter 10.

9.7. CONFIDENCE MEASURES: MEASURING THE RELIABILITY

One of the most critical components in a practical speech recognition system is a reliable *confidence measure*. With an accurate confidence measure for each recognized word, the conversational back end can repair potential speech recognition errors, can reject out-of-vocabulary words, and can identify key words (perform word spotting) that are relevant to the back end. In a speaker-dependent or speaker-adaptive system, the confidence measure can help user enrollment (to eliminate mispronounced words). It is also critical for unsupervised speaker adaptation, allowing selective use of recognition results so that transcriptions with lower confidence can be discarded for adaptation.

In theory, an accurate estimate of $P(\mathbf{W} \mid \mathbf{X})$, the posterior probability, is itself a good confidence measure for word $\mathbf{W}$ given the acoustic input $\mathbf{X}$. Most practical speech recognition systems simply ignore $P(\mathbf{X})$, as it is a constant in evaluating $P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W}) / P(\mathbf{X})$ across different words. $P(\mathbf{W} \mid \mathbf{X})$ can be expressed:

$$P(\mathbf{W} \mid \mathbf{X}) = \frac{P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})}{P(\mathbf{X})} = \frac{P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})}{\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})} \tag{9.34}$$

Equation (9.34) essentially provides a solid framework for measuring confidence levels. It is the ratio between the score for the word hypothesis $P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$ and the acoustic probability $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$. In the sections that follow we discuss a number of ways to model and use such a ratio in practical systems.

9.7.1. Filler Models

You can compute $P(\mathbf{X})$ in Eq. (9.34) with a general-purpose recognizer. It should be able to recognize anything such that it can *fill the holes* of the grammar in the normal speech recognizer. The filler model has various forms [7, 63]. One of the most widely used is the so-called all-phone network, in which all the possible phonetic and nonspeech HMMs are connected to each other, and with which any word sequence can be recognized.

In addition to evaluating $P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$ as needed in normal speech recognition, a separate decoding process is used to evaluate $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X} \mid \mathbf{W})$. Here $\mathbf{W}$ is either a phonetic or a word model. You can also apply phonetic n-gram probabilities that are derived from a lexicon targeted for possible new words. The best path from the all-phone network is compared with the best path from the normal decoder. The ratio between the two, as expressed in Eq. (9.34), is used to measure the confidence for either word or phone. In the decoding process (see Chapters 12 and 13), you can accumulate the phonetic ratio derived from Eq. (9.34) on a specific word. If the accumulative $P(\mathbf{W} \mid \mathbf{X})$ for the word is less than a predetermined threshold, the word is rejected as either a new word or a nonspeech event.

Both context-independent and context-dependent phonetic models can be used for the fully connected network. When context-dependent phonetic models are used, you need to make sure that only correct contextual phonetic connections are made. Although context-dependent models offer significant improvement for speech recognition, the filler phonetic network seems to be insensitive to context-dependency in empirical experiments.

There are *word-spotting* applications that need to spot just a small number of key words. You can use the filler models described here for word spotting. You can also build antiword models trained with all the data that are not associated with the key words of interest. Empirical experiments indicate that large-vocabulary speech recognition is the most suitable choice for word spotting. You can use a general-purpose n-gram (see Chapter 11) to generate recognition results and identify needed key words from the word lattice. This is because a large-vocabulary system provides a better estimate of $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X}\,|\,\mathbf{W})$ with a more accurate language model probability. In practice, we don't need to use all the hypotheses to compute $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X}\,|\,\mathbf{W})$. Instead, n-best lists and scores [40] can be used to provide an effective estimate of $\sum_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X}\,|\,\mathbf{W})$.

9.7.2. Transformation Models

To determine the confidence level for each word, subword confidence information is often helpful. Different phones have different impacts on our perception of words. The weight for each subword confidence score can be optimized from the real training data. If a word w has N phones, we can compute the confidence score of the word as follows:

$$CS(w) = \sum_{i=1}^{N} \wp_i(x_i)/N \tag{9.35}$$

where $CS(w)$ is the confidence score for word w, x_i is the confidence score for subword unit i in word w, and $\wp_i$ is the mapping function that may be tied across a number of subword units. The transformation function can be defined as:

$$\wp_i(x) = ax + b \tag{9.36}$$

We can use discriminative training to optimize the parameters a and b, respectively. A cost function can be defined as a sigmoid function of $CS(w)$. As shown in Figure 9.12, the optimal transformation parameters vary substantially across different phones. The weight for consonants is also typically larger than that of vowels.

The transformation function can be context dependent. Figure 9.13 illustrates the ROC curve of the context-dependent transformation model in comparison with the corresponding phonetic filler model. The filler model essentially has a uniform weight across all the phones in a given word. The estimated transformation model has 15–40% false-acceptance error reduction at various fixed false-rejection rates. The false-acceptance rate of the transformation model is consistently lower than that of the filler model [63].

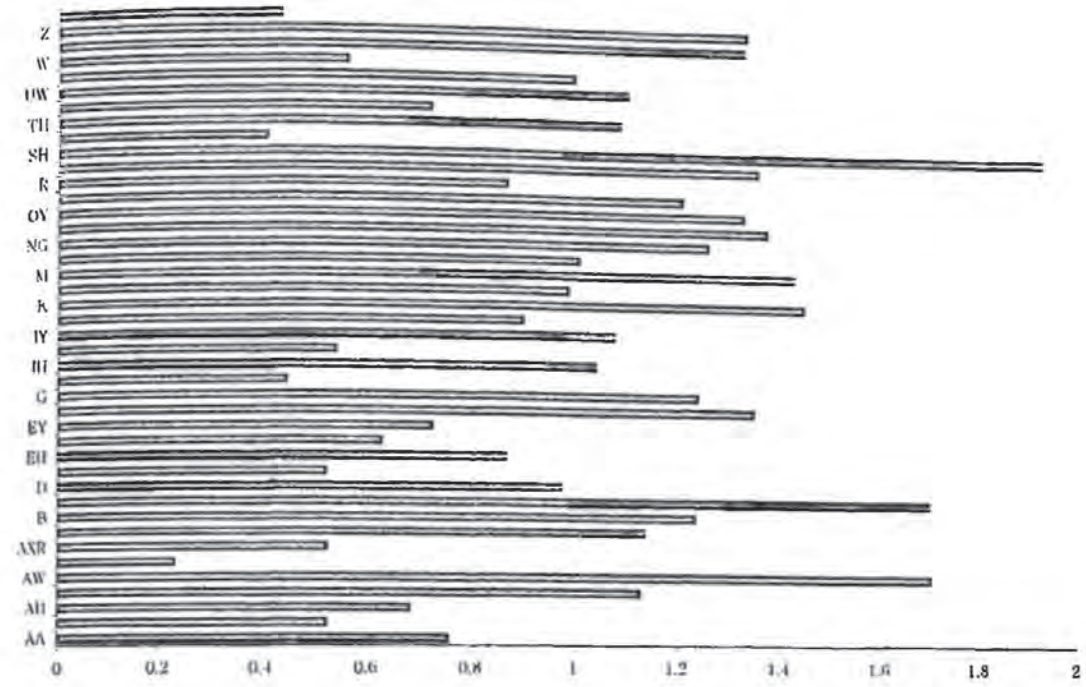


Figure 9.12 Transformation parameter a for each context-independent phone class. The weight of consonants is typically larger than that of vowels [63].

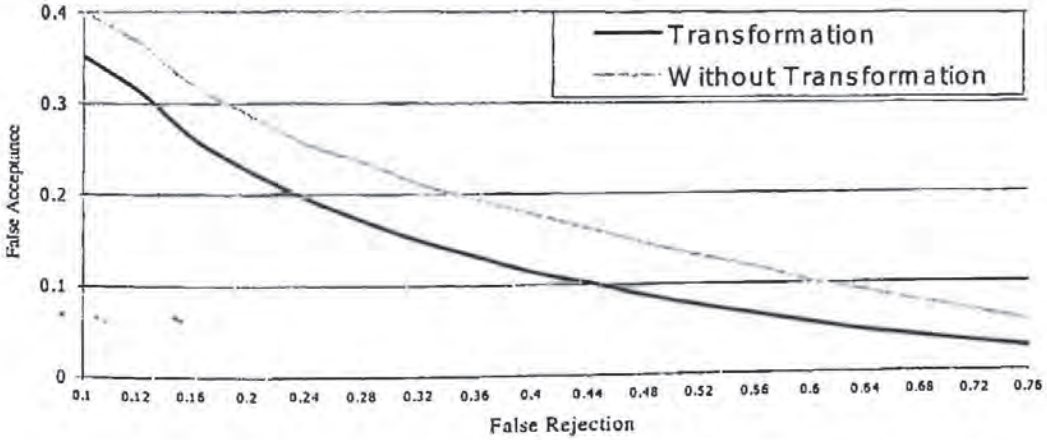


Figure 9.13 The ROC curve of phonetic filler models with and without optimal feature transformation [63].

9.7.3. Combination Models

In practical systems, there are a number of features you can use to improve the performance of confidence measures. For example, the following features are helpful:

- Word stability ratio from different language model weights (*WrdStabRatio*). This is obtained by applying different language weights to see how stably each word shows up in the recognition *n*-best list.
- Logarithm of the average number of active words around the ending frame of the word (*WrdCntEnd*).
- Acoustic score per frame within the word normalized by the corresponding active senone scores (*AscoreSen*).
- Logarithm of the average number of active words within the word (*WrdCntW*).
- Acoustic score per frame within the word normalized by the phonetic filler model (*AscoreFiller*).
- Language model score (*LMScore*).
- Language model back-off (trigram, bigram, or unigram hit) for the word (*LMBackOff*).
- Logarithm of the average number of active states within the word (*StateCnt*).
- Number of phones in the word (*Nphones*).
- Logarithm of the average number of active words around the beginning frame of the word (*WrdCntBeg*).
- Whether the word has multiple pronunciations (*Mpron*).
- Word duration (*WordDur*).

To clarify each feature's relative importance, Table 9.6 shows its linear correlation coefficient against the correct/incorrect tag for each word in the training set. Word stability ratio (*WrdStabRatio*) has the largest correlation value.

Several kinds of classifiers can be used to compute the confidence scores. Previous research has shown that the difference between classifiers, such as linear classifiers, generalized linear models, decision trees, and neural networks, is insignificant. The simplest linear classifier based on discriminative training performs well in practice. As some features are highly correlated, you can iteratively remove features to combat the curse of dimensionality. The combination model can have up to 40–80% false-acceptance error reduction at fixed false-rejection rate in comparison to the single-feature approach.

Table 9.6 Correlation coefficients of several features against correct/incorrect tag.

Feature	Correlation
WrdStabRatio	0.590
WrdCntW	−0.223
LMBackOff	0.171
AscoreSen	0.250
LMScore	0.175
Nphones	0.091
WordDur	0.012
WrdCntEnd	−0.321
AscoreFiller	0.219
StateCnt	−0.155
Mpron	0.057
WrdCntBeg	−0.067

9.8. OTHER TECHNIQUES

In addition to HMMs, a number of interesting alternative techniques are being actively investigated by researchers. We briefly review two promising methods here.

9.8.1. Neural Networks

You have seen both single-layer and multilayer neural nets in Chapter 4 for dealing with static patterns. In dealing with nonstationary signals, you need to address how to map an input sequence properly to an output sequence when two sequences are not synchronous, which should include proper alignment, segmentation, and classification. The basic neural networks are not well equipped to address these problems in a unified way.

Recurrent neural networks have an internal state that is a function of the current input and the previous internal state. A number of them use time-step delayed recurrent loops on the hidden or output units of a feedforward network, as discussed in earlier chapters. For sequences of finite numbers of delays, we can transform these networks into equivalent feedforward networks by unfolding them over the time period. They can be trained with the standard back propagation procedure, with the following modifications:

- The desired outputs are functions of time, and error functions have to be computed for every copy of the output layer. This requires the selection of an appropriate time-dependent target function, which is often difficult to define.
- All copies of the unfolded weights are constrained to be identical during the training. We can compute the correction terms separately for each weight and use the average to update the final estimate.

In most of these networks, you can have a partially recurrent network that has feedback of the hidden and output units to the input layer. For example, the feedforward network can be used in a set of local feedforward connections with one time-step delay. These networks are usually implemented by extending the input field with additional feedback units containing both the hidden and output values generated by the preceding input. You can encode the past nonstationary information that is often required to generate the correct output, given the current input, as illustrated in Figure 9.14.

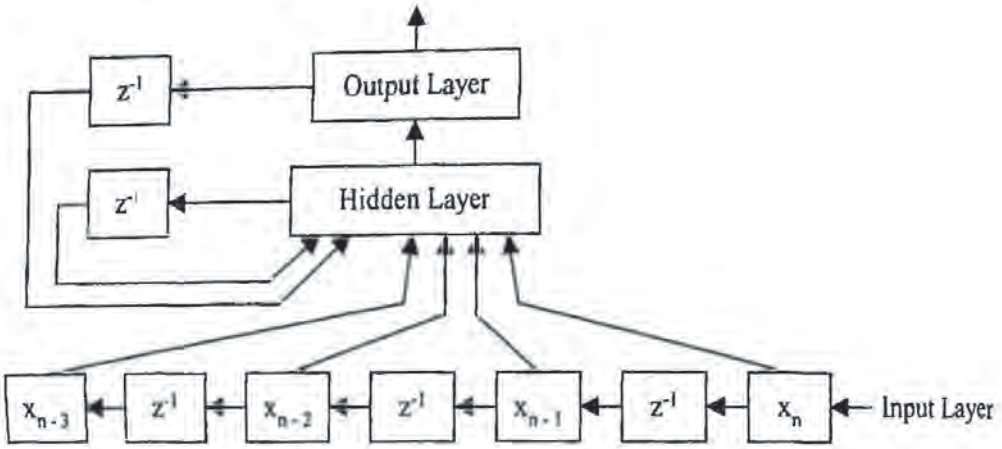


Figure 9.14 A recurrent network with contextual inputs, hidden vector feedback, and output vector feedback.

One of the popular neural networks is the *Time Delay Neural Network* (TDNN) [105]. Like static networks, the TDNN can be trained to recognize a sequence of predefined length (defined by the width of the input window). The activation in the hidden layer is computed from the current and multiple time-delayed values of the preceding layer, and the output units are activated only when a complete speech segment has been processed. A typical TDNN is illustrated in Figure 9.15. The TDNN has been successfully used to classify presegmented phonemes.

All neural networks have been shown to yield good performance for small-vocabulary speech recognition. Sometimes they are better than HMMs for short, isolated speech units. By recurrence and the use of temporal memory, they can perform some kind of integration over time. It remains a challenge for neural networks to demonstrate that they can be as effective as HMMs for dealing with nonstationary signals, as is often required for large-vocabulary speech recognition.

To deal with continuous speech, the most effective solution is to integrate neural nets with HMMs [91, 113]. The neural network can be used as the output probabilities to replace the Gaussian mixture densities. Comparable results can be obtained with the integrated approach. These HMM output probabilities could be estimated by applying the Bayes' rule to the output of neural networks that have been trained to classify HMM state categories. The neural networks can consist either of a single large trained network or of a group of separately trained small networks [21, 31, 75, 90].

A number of techniques have been developed to improve the performance of training these networks. Training can be embedded in an EM-style process. For example, dynamic programming can be used to segment the training data. The segmented data are then used to retrain the network. It is also possible to have Baum-Welch style training [14, 42].

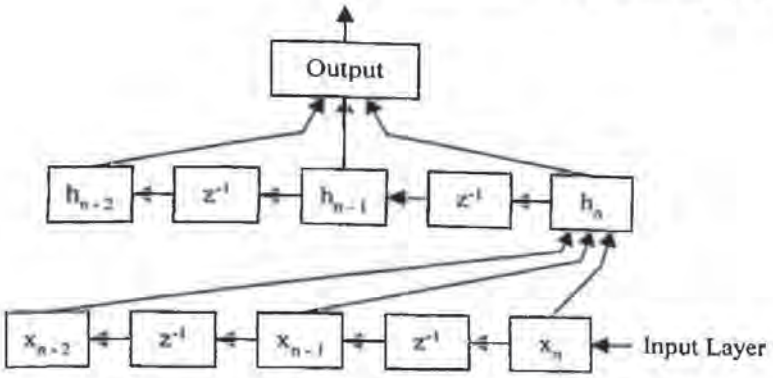


Figure 9.15 A time-delay neural network (TDNN), where the box h_t denotes the hidden vector at time t, the box x_t denotes the input vector at time t, and the box z^{-1} denotes a delay of one sample.

9.8.2. Segment Models

As discussed in Chapter 8, the HMM *output-independence assumption* results in a piecewise stationary process within an HMM state. Although the nonstationary speech may be modeled sufficiently with a large number of states, the states in which salient speech features are present are far from stationary [25, 99]. While the use of time-derivative features (e.g., delta and/or delta-delta features) alleviates these limitations, the use of such longer-time-span features may invalidate the conditional independence assumption.

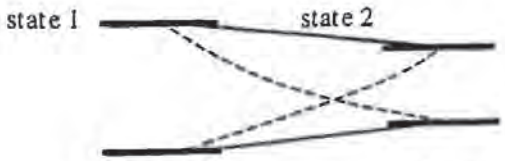


Figure 9.16 Diagram illustrating that HMM's output observation can hop between two unexpected quasi-stationary states [46].

The use of Gaussian mixtures for continuous or semicontinuous HMMs, as described in Chapter 8, could introduce another potential problem, where arbitrary transitions among the Gaussian mixture components between adjacent HMM states are allowed [59]. Figure 9.16 illustrates two HMM states with two mixture components. The solid lines denote the valid trajectories actually observed in the training data. However, in modeling these two trajectories, the Gaussian mixtures inadvertently allow two *phantom* trajectories, shown in

dashed lines in Figure 9.16, because no constraint is imposed on the mixture transitions across the state. It is possible that such phantom trajectories degrade recognition performance, because the models can overrepresent speech signals that should be modeled by other acoustic units. Segment models can alleviate such HMM modeling deficiencies [77, 79].

In the standard HMM, the output probability distribution is modeled by a quasi-stationary process, i.e.,

$$P(\mathbf{x}_1^L \mid s) = \prod_{l=1}^{L} b_s(\mathbf{x}_l) \tag{9.37}$$

For the segment model (SM), the output observation distribution is modeled by two stochastic processes:

$$P(\mathbf{x}_1^L \mid s) = P(\mathbf{x}_1^L \mid s, L)P(L \mid s) \tag{9.38}$$

The first term of Eq. (9.38) is no longer decomposable in the absence of the output-independence assumption. The second term is similar to the duration model described in Chapter 8. In contrast to the HMM whose quasi-stationary process for each state s generates one frame $\mathbf{x}_l$, a state in a segment model can generate a variable-length observation sequence $\{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_L\}$ with random length L.

Since the likelihood evaluation of segment models cannot be decomposed, the computation of the evaluation is not shareable between different segments (even for the case where two segments differ only by one frame). This results in a significant increase in computation for both training and decoding [77]. In general, the search state space is increased by a factor of $L_{\max}$, the maximum segment duration. If segment models are used for phone segments, $L_{\max}$ could be as large as 60. On top of this large increase in search state space, the evaluation of segment models is usually an order of magnitude more expensive than for HMM, since the evaluation involves several frames. Thus, the segment model is often implemented in a multipass search framework, as described in Chapter 13.

Segment models have produced encouraging performance for small-vocabulary or isolated recognition tasks [25, 44, 79]. However, their effectiveness on large-vocabulary continuous speech recognition remains an open issue because of necessary compromises to reduce the complexity of implementation.

9.8.2.1. Parametric Trajectory Models

Parametric trajectory models [25, 37] were first proposed to model a speech segment with curve-fitting parameters. They approximate the D-dimensional acoustic observation vector $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ by a polynomial function. Specifically, the observation vector $\mathbf{x}_t$ can be represented as

$$\mathbf{x}_t = \mathbf{C} \times F_t + \mathbf{e}_t(\Sigma) = \begin{pmatrix} c_1^0 & c_1^1 & \cdots & c_1^N \\ c_2^0 & c_2^1 & \cdots & c_2^N \\ \vdots & \vdots & \vdots & \vdots \\ c_D^0 & c_D^1 & c_D^2 & c_D^N \end{pmatrix} \begin{pmatrix} f_0(t) \\ f_1(t) \\ \vdots \\ f_N(t) \end{pmatrix} + \mathbf{e}_t(\Sigma) \tag{9.39}$$

where the matrix $\mathbf{C}$ is the trajectory parameter matrix, F_t is the family of N^{th}-order polynomial functions, and $\mathbf{e}_t(\Sigma)$ is the residual fitting error. Equation (9.39) can be regarded as modeling the time-varying mean in the output distribution for an HMM state. To simplify computation, the distribution of the residual error is often assumed to be an independent and identically distributed random process with a normal distribution $N(0, \Sigma)$. To accommodate diverse durations for the same segment, the relative linear time sampling of the fixed trajectory is assumed [37].

Each segment M is characterized by a trajectory parameter matrix $\mathbf{C}_m$ and covariance matrix Σ_m. The likelihood for each frame can be specified [46] as

$$P(\mathbf{x}_t \mid \mathbf{C}_m, \Sigma_m) = \frac{\exp\left\{-tr\left[(\mathbf{x}_t - \mathbf{C}_m F_t)\Sigma_m^{-1}(\mathbf{x}_t - \mathbf{C}_m F_t)^t\right]/2\right\}}{(2\pi)^{D/2} \mid \Sigma_m \mid^{1/2}} \tag{9.40}$$

If we let $\mathbf{F}_T = (F_0, F_1, \cdots, F_{T-1})^t$, then the likelihood for the whole acoustic observation vector can be expressed as

$$P(\mathbf{X} \mid \mathbf{C}_m, \Sigma_m) = \frac{\exp\left\{-tr\left[(\mathbf{X} - \mathbf{C}_m \mathbf{F}_T)\Sigma_m^{-1}(\mathbf{X} - \mathbf{C}_m \mathbf{F}_T)^t\right]/2\right\}}{(2\pi)^{DT/2} \mid \Sigma_m \mid^{T/2}} \tag{9.41}$$

Multiple mixtures can also be applied to SM. Suppose segment M is modeled by K trajectory mixtures. The likelihood for the acoustic observation vector $\mathbf{X}$ becomes

$$\sum_{k=1}^{K} w_k p_k(\mathbf{X} \mid \mathbf{C}_k, \Sigma_k) \tag{9.42}$$

Hon et al. [47] showed that only a handful of target trajectories are needed for speaker-independent recognition, in contrast to the many mixtures required for continuous Gaussian HMMs. This should support the phantom-trajectory argument involved in Figure 9.16.

The estimation of segment parameters can be accomplished by the EM algorithm described in Chapter 4. Assume a sample of L observation segments $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_L$, with corresponding duration $T_1, T_2, \cdots, T_L$, are generated by the segment model M. The MLE formulae using the EM algorithm are:

$$\gamma_k^i = \frac{w_k p_k(\mathbf{X}_i \mid \mathbf{C}_k, \Sigma_k)}{P(\mathbf{X}_i \mid \Phi_m)} \tag{9.43}$$

$$\hat{w}_k = \frac{1}{L}\sum_{i=1}^{L}\frac{w_k p_k(\mathbf{X}_i \mid \mathbf{C}_k, \Sigma_k)}{P(\mathbf{X}_i \mid \Phi_m)} \tag{9.44}$$

$$\hat{\mathbf{C}}_k = \left[\sum_{i=1}^{L}\gamma_k^i \mathbf{X}_i \mathbf{F}_{T_i}^t\right] / \left[\sum_{i=1}^{L}\gamma_k^i \mathbf{F}_{T_i}\mathbf{F}_{T_i}^t\right] \tag{9.45}$$

$$\hat{\Sigma}_k = \sum_{i=1}^{L}\gamma_k^i (\mathbf{X}_i - \mathbf{C}_k\mathbf{F}_{T_i})(\mathbf{X}_i - \mathbf{C}_k\mathbf{F}_{T_i})^t \Big/ \sum_{i=1}^{L}\gamma_k^i T_i \tag{9.46}$$

Parametric trajectory models have been successfully applied to phone classification [25, 46] and word spotting [37], and offer a modestly improved performance over HMMs.

9.8.2.2. Unified Frame- and Segment-Based Models

The strengths of the HMM and the segment-model approaches are complementary. HMMs are very effective in modeling the subtle details of speech signals by using one state for each quasi-stationary region. However, the transitions between quasi-stationary regions are largely neglected by HMMs because of their short durations. In contrast, segment models are powerful in modeling the transitions and longer-range speech dynamics, but might need to give up the detailed modeling to assure trainability and tractability. It is possible to have a unified framework to combine both methods [47].

In the unified complementary framework, the acoustic model $p(\mathbf{X}_1^T \mid \mathbf{W})$ can be considered as two joint hidden processes, as in the following equation:

$$\begin{aligned} p(\mathbf{X} \mid \mathbf{W}) &= \sum_{\mathbf{q}^h}\sum_{\mathbf{q}^s} p(\mathbf{X}, \mathbf{q}^h, \mathbf{q}^s \mid \mathbf{W}) \\ &= \sum_{\mathbf{q}^h}\sum_{\mathbf{q}^s} p(\mathbf{X} \mid \mathbf{q}^h, \mathbf{q}^s) p(\mathbf{q}^s \mid \mathbf{q}^h) p(\mathbf{q}^h \mid \mathbf{W}) \end{aligned} \tag{9.47}$$

where $\mathbf{q}^h$ represents the hidden process of the HMM and $\mathbf{q}^s$, the segment model. The conditional probability of the acoustic signal $p(\mathbf{X} \mid \mathbf{q}^s, \mathbf{q}^h)$ can be further decomposed into two separate terms:

$$p(\mathbf{X} \mid \mathbf{q}^s, \mathbf{q}^h) = p(\mathbf{X} \mid \mathbf{q}^h) p(\mathbf{X} \mid \mathbf{q}^s)^a \tag{9.48}$$

where a is a constant that is called *segment-model weight*. The first term is the contribution from normal frame-based HMM evaluation. We further assume for the second term that recognition of segment units can be performed by detecting and decoding a sequence of salient events in the acoustic stream that are statistically independent. In other words,

$$p(\mathbf{X} \mid \mathbf{q}^s) = \prod_i p(\mathbf{X}_i \mid q_i^s) \tag{9.49}$$

where $\mathbf{X}_i$ denotes the i^{th} segment.

We assume that the phone sequence and the phone boundaries hypothesized by HMMs and segment models agree with each other. Based on the independent-segment assumption, this leads to a segment duration model as

$$p(\mathbf{q}^s \mid \mathbf{q}^h) = \prod_i p(t_i, t_{i+1} - 1 \mid \mathbf{X}_i) \tag{9.50}$$

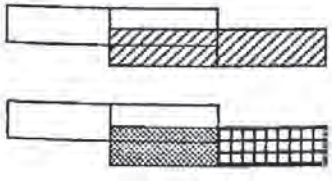
By treating the combination as a hidden-data problem, we can apply the decoding and iterative EM reestimation techniques here. This unified framework enables both frame- and segment-based models to *jointly* contribute to optimal segmentations, which leads to more efficient pruning during the search. The inclusion of the segment models does not require massive revisions in the decoder, because the segment model scores can be handled in the same manner as the language model scores; whereas the segment evaluation is performed at each segment boundary.

Since subphonetic units are often used in HMMs to model the detailed quasi-stationary speech region, the segment units should be used to model long-range transition. As studies have shown that phone transitions play an essential role in humans' perception, the phone-pair segment unit that spans over two adjacent phones can be used [47]. Let e_i and t_i denote the phone and the starting time of the i^{th} segment, respectively. For a phone-pair (e_{i-1}, e_i) segment between t_i and t_{i+1}, the segment likelihood can be computed as follows:

$$p(\mathbf{X}_i \mid q_i^s) = p(\mathbf{x}_{t_{i-1}}^{t_i-1} \mid e_{i-1}, e_i) \tag{9.51}$$

Rather than applying segment evaluation for every two phones, an *overlapped evaluation* scheme can be used, as shown in Figure 9.17 (a), where a phone-pair segment model evaluation is applied at each phone boundary. The overlapped evaluation implies that each phone is evaluated twice in the overall score. Most importantly, the overlapped evaluation places constraints on overlapped regions to assure consistent trajectory transitions. This is an important feature, as trajectory mixtures prohibit *phantom* trajectories within a segment unit, but there is still no mechanism to prevent arbitrary trajectory transitions between adjacent segment units.

Some phone-pairs might not have sufficient training data. Units containing silence might also have obscure trajectories due to the arbitrary duration of silence. As a result, a



(a) phone-pair segment models

(b) two phone (monophone or gen. triphone) segment models

Figure 9.17 Overlapped evaluation using (a) a phone-pair segment model, or (b) back-off to two phone units when the phone-pair (e_{i-1}, e_i) segment model does not exist [47].

phone-pair unit (e_{i-1}, e_i) can be backed off with two phone units as shown in Figure 9.17 (b). The phone units can be context independent or context dependent [46]. Thus, the back-off segment-model evaluation becomes:

$$p(\mathbf{X}_i \mid q_i^s) = \beta * p(\mathbf{x}_{i,-1}^{t_i} \mid e_{i-1}) p(\mathbf{x}_{i,}^{t_{i+1}} \mid e_i) \tag{9.52}$$

where β is the back-off weight, generally smaller than 1.0. The use of back-off weight has the effect of giving more preference to phone-pair segment models than to two-phone-based back-off segment models.

The phone-pair segment model outperformed the phone-pair HMM by more than 20% in a phone-pair classification experiment [46]. The unified framework achieved about 8% word-error-rate reduction on the WSJ dictation task in comparison to the HMM-based Whisper [47].

9.9. CASE STUDY: WHISPER

Microsoft's Whisper engine offers general-purpose speaker-independent continuous speech recognition [49]. Whisper can be used for command and control, dictation, and conversational applications. Whisper offers many features such as continuous speech recognition, speaker-independence with adaptation, and dynamic vocabulary. Whisper has a unified architecture that can be scaled to meet different application and platform requirements.

The Whisper system uses MFCC representations (see Chapter 6) and both first- and second-order delta MFCC coefficients. Two-mean cepstral normalization discussed in Chapter 10 is used to eliminate channel distortion for improved robustness.

The HMM topology is a three-state left-to-right model for each phone. Senone models discussed in Section 9.4.3 are derived from both inter- and intraword context-dependent phones. The generic shared density function architecture can support either semicontinuous or continuous density hidden Markov models.

The SCHMM has a multiple-feature front end. Independent codebooks are built for the MFCC, first-order delta MFCC, second-order delta MFCC, and power and first and second power, respectively. Deleted interpolation is used to interpolate output distributions of context-dependent and context-independent senones. All codebook means and covariance matrices are reestimated together with the output distributions except the power covariance matrices, which are fixed.

The overall senone models can reduce the error rate significantly in comparison to the triphone or clustered triphone model. The shared Markov state also makes it possible to use continuous-density HMMs efficiently for large-vocabulary speech recognition. When a sufficient amount of training data becomes available, the best performance is obtained with the continuous-density mixture HMM. Each senone has 20 mixtures, albeit such an error reduction came at the cost of significantly increased computational complexity.

We can further generalize sharing to the level of each individual Gaussian probability density function. Each Gaussian function is treated as the basic unit to be shared across any Markov state. At this extreme, there is no need to use senones or shared states any more, and

the shared probability density functions become the acoustic kernels that can be used to form any mixture function for any Markov state with appropriate mixture weights. Parameter sharing is, thus, advanced from a phone unit to a Markov state unit (senones) to a density component unit.

Regarding lexicon modeling, most words have one pronunciation in the lexicon. For words that are not in the dictionary, the LTS conversion is based on the decision tree that is trained from the existing lexicon. For the purpose of efficiency, the dictionary is used to store the most frequently used words. The LTS is only used for new words that need to be added on the fly.

For speaker adaptation, the diagonal variances and means are adapted using the MAP method. Whisper also uses MLLR to modify the mean vectors only. The MLLR classes are phone dependent. The transition probabilities are context independent and they are not modified during the adaptation stage.

The language model used in Whisper can be either the trigram or the context-free grammar. The difference is largely related to the decoder algorithm, as discussed in Chapter 13. The trigram lexicon has the 60,000 most-frequent words extracted from a large text corpus. Word selection is based on both the frequency and the word's part-of-speech information. For example, verbs and the inflected forms have a higher weight than proper nouns in the selection process.

Whisper's overall word recognition error rate for speaker-independent continuous speech recognition is about 7% for the standard DARPA business-news dictation test set. For isolated dictation with similar materials, the error rate is less than 3%. If speaker-dependent data are available, it can further reduce the error rate by 15–30%, with less than 30 minutes' speech from each person. The performance can be obtained real-time on today's PC systems.

9.10. Historical Perspective and Further Reading

The first machine to recognize speech was a commercial toy named Radio Rex manufactured in the 1920s. Fueled by increased computing resources, acoustic-phonetic modeling has progressed significantly since then. Relative word error rates have been reduced by 10% every year, as illustrated in Figure 9.18, thanks to the use of HMMs, the availability of large speech and text corpora, the establishment of standards for performance evaluation, and advances in computer technology. Before the HMM was established as the standard, there were many competing techniques, which can be traced back to the 1950s. Gold and Morgan's book provides an excellent historical perspective [38].

The HMM is powerful in that, with the availability of training data, the parameters of the model can be estimated and adapted automatically to give optimal performance. There are many HMM-based state-of-the-art speech recognition systems [1, 12, 27, 34, 49, 55, 72, 73, 93, 108, 109, 112]. Alternatively, we can first identify speech segments, then classify the segments and use the segment scores to recognize words. This approach has produced competitive recognition performance that is similar to HMM-based systems in several small- to medium-vocabulary tasks [115].

Speech recognition systems attempt to model the sources of variability in several ways. At the level of signal representation, in addition to MFCC, researchers have developed representations that emphasize perceptually important speaker-independent features of the signal, and deemphasize speaker-dependent characteristics [43]. Other methods based on linear discriminant analysis to improve class separability [28, 54] and speaker normalization transformation to minimize speaker variations [51, 67, 86, 106, 107, 114] have achieved limited success. Linear discriminant analysis can be traced back to *Fisher's linear discriminant* [30], which projects a dimensional vector onto a single line that is oriented to maximize class separability. Its extension to speech recognition can be found in [65].

At the level of acoustic-phonetic modeling, we need to provide an accurate distance measure of the input feature vectors against the phonetic or word models from the signal-processing front end. Before the HMM was used, the most successful acoustic-phonetic model was based on the speech template where the feature vectors are stored as the model and dynamic-programming-based time warping was used to measure the distance between the input feature vectors and the word or phonetic templates [88, 94]. The biggest problem for template-based systems is that they are not as trainable as HMMs, since it is difficult to generate a template that is as representative as all the speech samples we have for the particular units of interest.

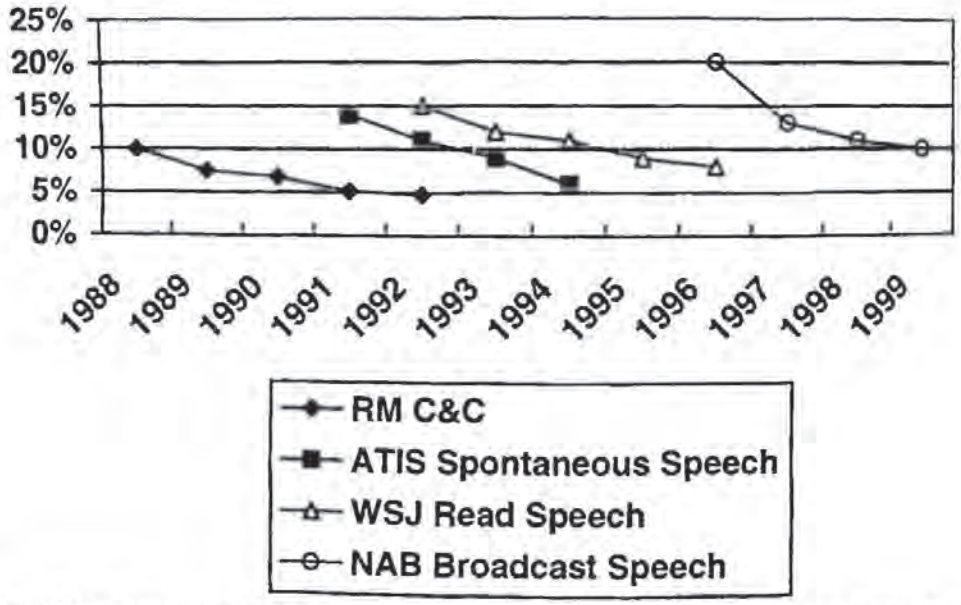


Figure 9.18 History of DARPA speech recognition word-error-rate benchmark evaluation results from 1988 to 1999. There are four major tasks: the Resource Management command and control task (RM C&C, 1000 words), the Air Travel Information System spontaneous speech understanding task (ATIS, 2000 words), the *Wall Street Journal* dictation task (WSJ, 20,000 words), and the Broadcast News Transcription Task (NAB, 60,000 words) [80-84].

Another approach that attracted many researchers is the knowledge-based one that originated from the Artificial Intelligence research community. This approach requires extensive knowledge engineering, which often led to many inconsistent rules. Due to the com-

plexity of speech recognition, rule-based approaches generally cannot match the performance of data-driven approaches such as HMMs, which can automatically extract salient rules from a large amount of training data [105].

Senones are now widely used in many state-of-the-art systems. Word models or allophone models can also be built by concatenation of basic structures made by states, transitions, and distributions such as *fenones* [8, 9] or *senones* [58].

Segment models, as proposed by Roucos and Ostendorf [79, 92], assume that each variable-length segment is mapped to a fixed number of representative frames. The resulting model is very similar to the HMM with a large number of states. Ostendorf published a comprehensive survey paper [77] on segment models. The parametric trajectory segment model was introduced by Deng et al. [25] and Gish et al. [37] independently. Gish's work is very similar to our description in Section 9.8.2.1, which is based on Hon et al. [46, 47], where the evaluation and estimation are more efficient because no individual polynomial fitting is required for likelihood computation. In addition to the phone-pair units described in this chapter, segment models have also been applied to phonetic units [25], subphonetic units [25], diphones [36], and syllables [78]. The dynamic model [24, 26] is probably the most aggressive attempt to impose a global transition constraint on the speech model. It uses the phonetic target theories on unobserved vocal-tract parameters, which are fed to an MLP to produce the observed acoustic data.

Today, it is not uncommon to have tens of thousands of sentences available for system training and testing. These corpora permit researchers to quantify the acoustic cues important for phonetic contrasts and to determine parameters of the recognizers in a statistically meaningful way. While many of these corpora were originally collected under the sponsorship of the U.S. Defense Advanced Research Projects Agency (ARPA) to spur human language technology development among its contractors [82], they have nevertheless gained international acceptance. Recognition of the need for shared resources led to the creation of the Linguistic Data Consortium (LDC)[5] in the United States in 1992 to promote and support the widespread development and sharing of resources for human language technology. The LDC supports various corpus development activities and distributes corpora obtained from a variety of sources. Currently, LDC distributes about twenty different speech corpora including those cited above, comprising many hundreds of hours of speech. The availability of a large body of data in the public domain, coupled with the specification of evaluation standards, has resulted in uniform documentation of test results, thus contributing to greater reliability in monitoring progress.

To further improve the performance of acoustic-phonetic models, we need a robust system so that performance degrades gracefully (rather than catastrophically) as conditions diverge from those under which it was trained. The best approach is likely to have systems continuously adapted to changing conditions (new speakers, microphone, task, etc.). Such adaptation can occur at many levels in systems, subword models, word pronunciations, language models, and so on. We also need to make the system portable, so that we can rapidly design, develop, and deploy systems for new applications. At present, systems tend to suffer

[5] http://www.cis.upenn.edu/ldc

significant degradation when moved to a new task. In order to retain peak performance, they must be trained on examples specific to the new task, which is time consuming and expensive. In the new task, system users may not know exactly which words are in the system vocabulary. This leads to a certain percentage of out-of-vocabulary words in natural conditions. Currently, systems lack a very robust method of detecting such out-of-vocabulary words. These words often are inaccurately mapped into the words in the system, causing unacceptable errors.

An introduction to all aspects of acoustic modeling can be found in *Spoken Dialogues with Computers* [76] and *Fundamentals of Speech Recognition* [87]. A good treatment of HMM-based speech recognition is given in [52, 60, 105]. Bourlard and Morgan's book [15] is a good introduction to speech recognition based on neural networks. There are a range of applications such as predictive networks that estimate each frame's acoustic vector, given the history [69, 104] and nonlinear transformation of observation vectors [13, 53, 101].

You can find tools to build acoustic models from Carnegie Mellon University's speech open source Web site.[6] This site contains the release of CMU's Sphinx acoustic modeling toolkit and documentation. A version of Microsoft's Whisper system can be found in the Microsoft Speech SDK.[7]

REFERENCES

[1] Abrash, V., *et al.*, "Acoustic Adaptation Using Nonlinear Transformations of HMM Parameters" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1996, Atlanta, pp. 729-732.

[2] Acero, A., *Acoustical and Environmental Robustness in Automatic Speech Recognition*, 1993, Boston, Kluwer Academic Publishers.

[3] Ahadi-Sarkani, S.M., *Bayesian and Predictive Techniques for Speaker Adaptation*, Ph. D. Thesis, 1996, Cambridge University, .

[4] Ahn, S., S. Kang, and H. Ko, "Effective Speaker Adaptations for Speaker Verification," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey, pp. 1081-1084.

[5] Alleva, F., *et al.*, "Can Continuous Speech Recognizers Handle Isolated Speech?," *Speech Communication*, 1998, **26**, pp. 183-189.

[6] Anastasakos, T., *et al.*, "A Compact Model for Speaker Adaptive Training," *Int. Conf. on Spoken Language Processing*, 1996, Philadelphia, pp. 1137-1140.

[7] Asadi, A., R. Schwartz, and J. Makhoul, "Automatic Modeling for Adding New Words to a Large-Vocabulary Continuous Speech Recognition System," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, pp. 305-308.

[8] Bahl, L.R., *et al.*, "Multonic Markov Word Models for Large Vocabulary Continuous Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(3), pp. 334-344.

[6] http://www.speech.cs.cmu.edu/sphinx/
[7] http://www.microsoft.com/speech

[9] Bahl, L.R., *et al.*, "A Method for the Construction of Acoustic Markov Models for Words," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(4), pp. 443-452.

[10] Bahl, L.R., *et al.*, "Automatic Phonetic Baseform Determination," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, pp. 173-176.

[11] Bahl, L.R., *et al.*, "Decision Trees for Phonological Rules in Continuous Speech," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, Canada, pp. 185-188.

[12] Bellegarda, J.R., *et al.*, "Experiments Using Data Augmentation for Speaker Adaptation," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, pp. 692-695.

[13] Bengio, Y., *et al.*, "Global Optimization of a Neural Network-Hidden Markov Model Hybrid," *IEEE Trans. on Neural Networks*, 1992, **3**(2), pp. 252-259.

[14] Bourlard, H., "A New Training Algorithm for Statistical Sequence Recognition with Applications to Transition-Based Speech Recognition," *IEEE Signal Processing Letters*, 1996, **3**, pp. 203-205.

[15] Bourlard, H. and N. Morgan, *Connectionist Speech Recognition - A Hybrid Approach*, 1994, Boston, MA, Kluwer Academic Publishers.

[16] Brown, P.F., *The Acoustic-Modeling Problem in Automatic Speech Recognition*, PhD Thesis in Computer Science Department, 1987, Carnegie Mellon University, Pittsburgh, PA.

[17] Campbell, J., "Speaker Recognition: A Tutorial," *Proc. of the IEEE*, 1997, **85**(9), pp. 1437-1462.

[18] Chesta, C., O. Siohan, and C.H. Lee, "Maximum a Posteriori Linear Regression for Hidden Markov Model Adaptation," *Eurospeech*, 1999, Budapest, pp. 211-214.

[19] Chou, W., "Maximum a Posteriori Linear Regression with Elliptically Symmetric Matrix Priors," *Eurospeech*, 1999, Budapest, pp. 1-4.

[20] Cohen, M., *Phonological Structures for Speech Recognition*, Ph.D. Thesis 1989, University of California, Berkeley.

[21] Cook, G. and A. Robinson, "Transcribing Broadcast News with the 1997 Abbot System," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle, WA, pp. 917-920.

[22] Cox, S., "Predictive Speaker Adaptation in Speech Recognition," *Computer Speech and Language*, 1995, **9**, pp. 1-17.

[23] Davis, S. and P. Mermelstein, "Comparison of Parametric Representations for Monosyllable Word Recognition in Continuously Spoken Sentences," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1980, **28**(4), pp. 357-366.

[24] Deng, L., "A Dynamic, Feature-based Approach to the Interface Between Phonology and Phonetics for Speech Modeling and Recognition," *Speech Communication*, 1998, **24**(4), pp. 299-323.

[25] Deng, L., *et al.*, "Speech Recognition Using Hidden Markov Models with Polynomial Regression Functions as Nonstationary States," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(4), pp. 507-520.

[26] Digalakis, V., *Segment-based Stochastic Models of Spectral Dynamics for Continuous Speech Recognition*, Ph.D. Thesis in *Electrical Computer System Engineering*, 1992, Boston University.

[27] Digalakis, V. and H. Murveit, "Genones: Optimizing the Degree of Mixture Tying in a Large Vocabulary Hidden Markov Model Based Speech Recognizer" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1994, Adelaide, Australia, pp. 537-540.

[28] Doddington, G.R., "Phonetically Sensitive Discriminants for Improved Speech Recognition," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, Glasgow, Scotland, pp. 556-559.

[29] Eichner, M. and M. Wolff, "Data-Driven Generation of Pronunciation Dictionaries in the German Verbmobil Project—Discussion of Experimental Results," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey, pp. 1687-1690.

[30] Fisher, R., "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, 1936, 7(1), pp. 179-188.

[31] Fritsch, J. and M. Finke, "ACID/HNN: Clustering Hierarchies of Neural Networks for Context-Dependent Connectionist Acoustic Modeling," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle, WA, pp. 505-508.

[32] Fukunaga, K., *Introduction to Statistical Pattern Recognition*, 2nd ed, 1990, Orlando, FL, Academic Press.

[33] Gales, M., "Cluster Adaptive Training for Speech Recognition," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia, pp. 1783-1786.

[34] Gauvain, J.L., L. Lamel, and M. Adda-Decker, "Developments in Continuous Speech Dictation using the ARPA WSJ Task," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, MI, pp. 65-68.

[35] Gauvain, J.L. and C.H. Lee, "Bayesian Learning of Gaussian Mixture Densities for Hidden Markov Models," *Proc. of the DARPA Speech and Natural Language Workshop*, 1991, Palo Alto, CA, pp. 272-277.

[36] Ghitza, O. and M.M. Sondhi, "Hidden Markov Models with Templates as Non-Stationary States: An Application to Speech Recognition," *Computer Speech and Language*, 1993, 7(2), pp. 101-120.

[37] Gish, H. and K. Ng, "A Segmental Speech Model with Applications to Word Spotting," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1993, Minneapolis, MN, pp. 447-450.

[38] Gold, B. and N. Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2000, New York, John Wiley.

[39] Greenberg, S., D. Ellis, and J. Hollenback, "Insights into Spoken Language Gleaned from Phonetic Transcription of the Switchboard Corpus," *Int. Conf. on Spoken Language Processing*, 1996, Philadelphia, PA, pp. addendum 24-27.

[40] Gunawardana, A., H.W. Hon, and L. Jiang, "Word-Based Acoustic Confidence Measures for Large-Vocabulary Speech Recognition," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia, pp. 791-794.

[41] Haeb-Umbach, R., "Investigations on Inter-Speaker Variability in the Feature Space," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1999, Phoenix, AZ.

[42] Hennebert, J., *et al.*, "Estimation of Global Posteriors and Forward-Backward Training of Hybrid HMM/ANN Systems," *Proc. of the Eurospeech Conf.*, 1997, Rhodes, Greece, pp. 1951-1954.

[43] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech," *Journal of the Acoustical Society of America*, 1990, 87(4), pp. 1738-1752.

[44] Holmes, W. and M. Russell, "Probabilistic-Trajectory Segmental HMMs," *Computer Speech and Language*, 1999, **13**, pp. 3-37.

[45] Hon, H.W. and K.F. Lee, "CMU Robust Vocabulary-Independent Speech Recognition System," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, pp. 889-892.

[46] Hon, H.W. and K. Wang, "Combining Frame and Segment Based Models for Large Vocabulary Continuous Speech Recognition," *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999, Keystone, CO.

[47] Hon, H.-W. and K. Wang, "Unified Frame and Segment Based Models for Automatic Speech Recognition," *Int. Conf. on Acoustic, Signal and Speech Processing*, 2000, Istanbul, Turkey, IEEE, pp. 1017-1020.

[48] Huang, X., *et al.*, "From Sphinx II to Whisper: Making Speech Recognition Usable" in *Automatic Speech and Speaker Recognition*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds. 1996, Norwell, MA, pp. 481-508, Kluwer Academic Publishers.

[49] Huang, X., *et al.*, "From Sphinx-II to Whisper - Make Speech Recognition Usable" in *Automatic Speech and Speaker Recognition*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds. 1996, Norwell, MA, Kluwer Academic Publishers.

[50] Huang, X. and K.-F. Lee, "On Speaker-Independent, Speaker-Dependent, and Speaker-Adaptive Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1993, 1(2), pp. 150-157.

[51] Huang, X.D., "Speaker Normalization for Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1992, San Francisco pp. 465-468.

[52] Huang, X.D., Y. Ariki, and M.A. Jack, *Hidden Markov Models for Speech Recognition*, 1990, Edinburgh, U.K., Edinburgh University Press.

[53] Huang, X.D., K. Lee, and A. Waibel, "Connectionist Speaker Normalization and its Applications to Speech Recognition," *IEEE Workshop on Neural Networks for Signal Processing*, 1991, New York, pp. 357-366.

[54] Hunt, M.J., *et al.*, "An Investigation of PLP and IMELDA Acoustic Representations and of Their Potential for Combination," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, pp. 881-884.

[55] Huo, Q., C. Chan, and C.-H. Lee, "On-Line Adaptation of the SCHMM Parameters Based on the Segmental Quasi-Bayes Learning for Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1996, 4(2), pp. 141-144.

[56] Hwang, M.Y., X. Huang, and F. Alleva, "Predicting Unseen Triphones with Senones," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1993, Minneapolis, pp. 311-314.

[57] Hwang, M.Y. and X.D. Huang, "Acoustic Classification of Phonetic Hidden Markov Models" in *Proc. of Eurospeech* 1991.

[58] Hwang, M.Y. and X.D. Huang, "Shared-Distribution Hidden Markov Models for Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1993, 1(4), pp. 414-420.

[59] Iyer, R., *et al.*, "Hidden Markov Models for Trajectory Modeling," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[60] Jelinek, F., *Statistical Methods for Speech Recognition*, 1998, Cambridge, MA, MIT Press.

[61] Jiang, L., H.W. Hon, and X. Huang, "Improvements on a Trainable Letter-to-Sound Converter," *Proc. of Eurospeech*, 1997, Rhodes, Greece, pp. 605-608.

[62] Jiang, L. and X. Huang, "Subword-Dependent Speaker Clustering for Improved Speech Recognition," *Int Conf. on Spoken Language Processing*, 2000, Beijing, China.

[63] Jiang, L. and X.D. Huang, "Vocabulary-Independent Word Confidence Measure Using Subword Features," *Int. Conf. on Spoken Language Processing*, 1998, Syndey, Australia.

[64] Kuhn, R., *et al.*, "Eigenvoices for Speaker Adaptation," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia, pp. 1771-1774.

[65] Kumar, N. and A. Andreou, "Heteroscedastic Discriminant Analysis and Reduced Rank HMMs for Improved Speech Recognition," *Speech Communication*, 1998, **26**, pp. 283-297.

[66] Lee, K.F., *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*, Ph.D. Thesis in *Computer Science Dept.* 1988, Carnegie Mellon University, Pittsburgh.

[67] Lee, L. and R. Rose, "Speaker Normalization Using Efficient Frequency Warping Procedures," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1996, Atlanta, GA, pp. 353-356.

[68] Leggetter, C.J. and P.C. Woodland, "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," *Computer Speech and Language*, 1995, **9**, pp. 171-185.

[69] Levin, E., "Word Recognition Using Hidden Control Neural Architecture," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1990, Albuquerque, NM, pp. 433-436.

[70] Lippmann, R.P., E.A. Martin, and D.P. Paul, "Multi-Style Training for Robust Isolated-Word Speech Recognition," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1987, Dallas, TX, pp. 709-712.

[71] Lucassen, J.M. and R.L. Mercer, "An Information-Theoretic Approach to the Automatic Determination of Phonemic Baseforms," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1984, San Diego, pp. 42.5.1-42.5.4.

[72] Lyu, R.Y., *et al.*, "Golden Mandarin (III) - A User-Adaptive Prosodic Segment-Based Mandarin Dictation Machine for Chinese Language with Very Large Vocabulary" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, MI, pp. 57-60.

[73] Matsui, T., T. Matsuoka, and S. Furui, "Smoothed N-best Based Speaker Adaptation for Speech Recognition" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1997, Munich, Germany, pp. 1015-1018.

[74] McDonough, J., *et al.*, "Speaker-Adapted Training on the Switchboard Corpus" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1997, Munich, Germany, pp. 1059-1062.

[75] Morgan, N. and H. Bourlard, *Continuous Speech Recognition: An Introduction to Hybrid HMM/Connectionist Approach*, in *IEEE Signal Processing Magazine*, 1995, pp. 25-42.

[76] Mori, R.D., *Spoken Dialogues with Computers*, 1998, London, Academic Press.

[77] Ostendorf, M., V.V. Digalakis, and O.A. Kimball, "From HMM's to Segment Models: a Unified View of Stochastic Modeling for Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1996, 4(5), pp. 360-378.

[78] Ostendorf, M. and K. Ross, "A Multi-Level Model for Recognition of Intonation Labels" in *Computing Prosody*, Y. Sagisaka, W.N. Campell, and N. Higuchi, eds., 1997, New York, pp. 291-308, Springer Verlag.

[79] Ostendorf, M. and S. Roukos, "A Stochastic Segment Model for Phoneme-Based Continuous Speech Recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1989, 37(1), pp. 1857-1869.

[80] Pallett, D., J.G. Fiscus, and J.S. Garofolo, "DARPA Resource Management Benchmark Test Results June 1990," in *Proc. of the DARPA Speech and Natural Language Workshop* 1990, Hidden Valley, PA, pp. 298-305, Pallett.

[81] Pallett, D., J.G. Fiscus, and J.S. Garofolo, "DARPA Resource Management Benchmark Test Results," *Proc. of the DARPA Speech and Natural Language Workshop*, 1991, Morgan Kaufmann Publishers, pp. 49-58.

[82] Pallett, D.S., *et al.*, "The 1994 Benchmark Tests for the ARPA Spoken Language Program" in *Proc. of the ARPA Spoken Language Technology Workshop*, 1995, Austin, TX, pp. 5-38.

[83] Pallett, D.S., *et al.*, "1997 Broadcast News Benchmark Test Results: English and Non-English," *Proc. of the Broadcast News Transcription and Understanding Workshop*, 1998, Landsdowne, Virginia, Morgan Kaufmann Publishers.

[84] Pallett, D.S., J.G. Fiscus, and M.A. Przybocki, "1996 Preliminary Broadcast News Benchmark Tests," *Proc. of the DARPA Speech Recognition Workshop*, 1997, Chantilly, VA, Morgan Kaufmann Publishers.

[85] Pereira, F., M. Riley, and R. Sproat, "Weighted Rational Transductions and Their Application to Human Language Processing," *Proc. of the ARPA Human Language Technology Workshop*, 1994, Plainsboro, NJ, pp. 249-254.

[86] Pye, D. and P.C. Woodland, "Experiments in Speaker Normalization and Adaptation for Large Vocabulary Speech Recognition," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1997, Munich, Germany, pp. 1047-1050.

[87] Rabiner, L.R. and B.H. Juang, *Fundamentals of Speech Recognition*, May, 1993, Prentice-Hall.

[88] Rabiner, L.R. and S.E. Levinson, "Isolated and Connected Word Recognition - Theory and Selected Applications," *IEEE Trans. on Communication*, 1981, COM-**29**(5), pp. 621-659.

[89] Riley, M. and A. Ljolje, eds. *Automatic Generation of Detailed Pronunciation Lexicons*, in Automatic Speech and Speaker Recognition, ed. C. Lee, F. Soong, and K. Paliwal, 1996, Kluwer Academic Publishers.

[90] Robinson, A., "An Application of Recurrent Nets to Phone Probability Estimation," *IEEE Trans. on Neural Networks*, 1994, **5**, pp. 298-305.

[91] Robinson, A.J., *et al.*, "A Neural Network Based, Speaker Independent, Large Vocabulary," *Proc. of the European Conf. on Speech Communication and Technology*, 1999, Berlin pp. 1941-1944.

[92] Roucos, S., *et al.*, "Stochastic Segment Modeling Using the Estimate-Maximize Algorithm," *Int. Conf. on Acoustic, Speech and Signal Processing*, 1988, New York, pp. 127-130.

[93] Sagisaka, Y. and L.S. Lee, "Speech Recognition of Asian Languages" in *Proc. IEEE Automatic Speech Recognition Workshop*, 1995, Snowbird, UT, pp. 55-57.

[94] Sakoe, H. and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1978, **26**(1), pp. 43-49.

[95] Saraclar, M. and S. Khudanpur, "Pronunciation Ambiguity vs. Pronunciation Variability in Speech Recognition," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey, pp. 1679-1682.

[96] Shinoda, K. and C. Lee, "Structural MAP Speaker Adaptation Using Hierarchical Priors," *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997, Santa Barbara, CA, pp. 381-388.

[97] Singh, R., B. Raj, and R. Stern, "Automatic Generation of Phone Sets and Lexical Transcriptions," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey, pp. 1691-1694.

[98] Siohan, O., C. Chesta, and C. Lee, "Joint Maximum a Posteriori Estimation of Transformation and Hidden Markov Model Parameters," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey, pp. 965-968.

[99] Siu, M., *et al.*, "Parametric Trajectory Mixtures for LVCSR," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[100] Sixtus, A., *et al.*, "Recent Improvements of the RWTH Large Vocabulary Speech Recognition System on Spontaneous Speech," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey, pp. 1671-1674.

[101] Sorenson, H., "A Cepstral Noise Reduction Multi-Layer Network," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, pp. 933-936.

[102] Sproat, R. and M. Riley, "Compilation of Weighted Finite-State Transducers from Decision Trees," *ACL-96*, 1996, Santa Cruz, pp. 215-222.

[103] Tajchman, G., E. Fosler, and D. Jurafsky, "Building Multiple Pronunciation Models for Novel Words Using Exploratory Computational Phonology," *Eurospeech*, 1995, pp. 2247-2250.

[104] Tebelskis, J. and A. Waibel, "Large Vocabulary Recognition Using Linked Predictive Neural Networks," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1990, Albuquerque, NM, pp. 437-440.

[105] Waibel, A.H. and K.F. Lee, *Readings in Speech Recognition*, 1990, San Mateo, CA, Morgan Kaufman Publishers.

[106] Watrous, R., "Speaker Normalization and Adaptation Using Second-Order Connectionist Networks," *IEEE Trans. on Neural Networks*, 1994, 4(1), pp. 21-30.

[107] Welling, L., S. Kanthak, and H. Ney, "Improved Methods for Vocal Tract Normalization," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1999, Phoenix, AZ.

[108] Wilpon, J.G., C.H. Lee, and L.R. Rabiner, "Connected Digit Recognition Based on Improved Acoustic Resolution," *Computer Speech and Language*, 1993, 7(1), pp. 15-26.

[109] Woodland, P.C., *et al.*, "The 1994 HTK Large Vocabulary Speech Recognition System," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, pp. 73-76.

[110] Wooters, C. and A. Stolcke, "Multiple-Pronunciation Lexical Modeling in a Speaker Independent Speech Understanding System," *Proc. of the Int. Conf. on Spoken Language Processing*, 1994, Yokohama, Japan, pp. 1363-1366.

[111] Young, S.J. and P.C. Woodland, "The Use of State Tying in Continuous Speech Recognition," *Proc. of Eurospeech*, 1993, Berlin pp. 2203-2206.

[112] Zavaliagkos, G., R. Schwartz, and J. Makhoul, "Batch, Incremental and Instantaneous Adaptation Techniques for Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, pp. 676-679.

[113] Zavaliagkos, G., *et al.*, "A Hybrid Segmental Neural Net/Hidden Markov Model System for Continuous Speech Recognition," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**, pp. 151-160.

[114] Zhan, P. and M. Westphal, "Speaker Normalization Based on Frequency Warping" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1997, Munich, Germany, pp. 1039-1042.

[115] Zue, V., *et al.*, "The MIT SUMMIT System: A Progress Report," *Proc. of DARPA Speech and Natural Language Workshop*, 1989, pp. 179-189.

EX-1018
U.S. Patent 10,755,699

CHAPTER 11

Language Modeling

Acoustic pattern matching, as discussed in Chapter 9, and knowledge about language are equally important in recognizing and understanding natural speech. Lexical knowledge (i.e., vocabulary definition and word pronunciation) is required, as are the syntax and semantics of the language (the rules that determine what sequences of words are grammatically well-formed and meaningful). In addition, knowledge of the pragmatics of language (the structure of extended discourse, and what people are likely to say in particular contexts) can be important to achieving the goal of spoken language understanding systems. In practical speech recognition, it may be impossible to separate the use of these different levels of knowledge, since they are often tightly integrated.

In this chapter we review the basic concept of Chomsky's formal language theory and the probabilistic language model. For the formal language model, two things are fundamental: the grammar and the parsing algorithm. The *grammar* is a formal specification of the permissible structures for the language. The *parsing* technique is the method of analyzing the sentence to see if its structure is compliant with the grammar. With the advent of bodies

of text (*corpora*) that have had their structures hand-annotated, it is now possible to generalize the formal grammar to include accurate probabilities. Furthermore, the probabilistic relationship among a sequence of words can be directly derived and modeled from the corpora with the so-called stochastic language models, such as *n*-gram, avoiding the need to create broad coverage formal grammars. Stochastic language models play a critical role in building a working spoken language system, and we discuss a number of important issues associated with them.

11.1. FORMAL LANGUAGE THEORY

In constructing a syntactic grammar for a language, it is important to consider the generality, the selectivity, and the understandability of the grammar. The *generality* and *selectivity* basically determine the range of sentences the grammar accepts and rejects. The *understandability* is important, since it is up to the authors of the system to create and maintain the grammar. For SLU systems described in Chapter 17, we need to have a grammar that covers and generalizes to most of the typical sentences for an application. The system also needs to distinguish the kind of sentences for different actions in a given application. Without understandability, it is almost impossible to improve a practical SLU system since it typically involves a large number of developers to maintain and refine the grammar.

The most common way of representing the grammatical structure of a sentence, "*Mary loves that person,*" is by using a tree, as illustrated in Figure 11.1. The node labeled *S* is the parent node of the nodes labeled *NP* and *VP* for noun phrase and verb phrase, respectively. The *VP* node is the parent node of node *V*—for verb. Each leaf is associated with the word

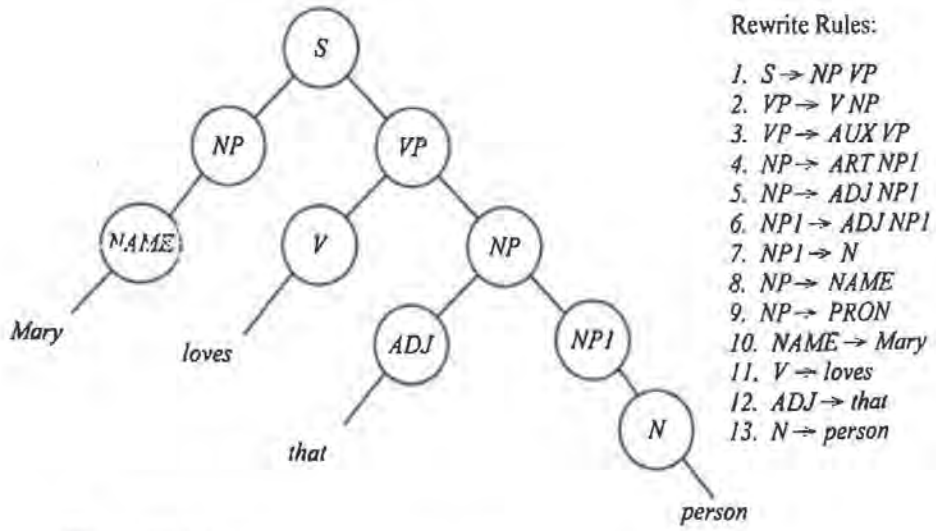


Figure 11.1 A tree representation of a sentence and its corresponding grammar.

in the sentence to be analyzed. To construct such a tree for a sentence, we must know the structure of the language so that a set of rewrite rules can be used to describe what tree structures are allowable. These rules, as illustrated in Figure 11.1, determine that a certain symbol may be expanded in the tree by a sequence of symbols. The grammatical structure helps in determining the meaning of the sentence. It tells us that *that* in the sentence modifies *person*. "Mary loves *that* person."

11.1.1. Chomsky Hierarchy

In Chomsky's *formal language theory* [1, 14, 15], a grammar is defined as $G = (V, T, P, S)$, where V and T are finite sets of *non-terminals* and *terminals*, respectively. V contains all the *non-terminal* symbols. We often use upper-case symbols to denote them. In the example discussed here, *S, NP, NP1, VP, NAME, ADJ, N*, and *V* are non-terminal symbols. The *terminal* set T contains *Mary, loves, that*, and *person*, which are often denoted with lower-case symbols. P is a finite set of *production (rewrite) rules*, as illustrated in the rewrite rules in Figure 11.1. S is a special non-terminal, called the *start symbol*.

The language to be analyzed is essentially a string of terminal symbols, such as "*Mary loves that person.*" It is produced by applying production rules sequentially to the start symbol. The production rule is of the form $\alpha \rightarrow \beta$, where α and β are arbitrary strings of grammar symbols V and T, and the α must not be empty. In formal language theory, four major languages and their associated grammars are hierarchically structured. They are referred to as the Chomsky hierarchy [1] as defined in Table 11.1. There are four kinds of automata that can accept the languages produced by these four types of grammars. Among these automata, the finite-state automaton is not only the mathematical device used to implement the regular grammar but also one of the most significant tools in computational linguistics. Variations of automata such as finite-state transducers, hidden Markov models, and n-gram models are important examples in spoken language processing.

These grammatical formulations can be compared according to their generative capacity, i.e., the range that the formalism can cover. While there is evidence that natural languages are at least weakly context sensitive, the context-sensitive requirements are rare in practice. The context-free grammar (CFG) is a very important structure for dealing with both machine language and natural language. CFGs are not only powerful enough to describe most of the structure in spoken language,[1] but also restrictive enough to have efficient parsers to analyze natural sentences. Since CFGs offer a good compromise between parsing efficiency and power in representing the structure of the language, they have been widely applied to natural language processing. Alternatively, regular grammars, as represented with a finite-state machine, can be applied to more restricted applications. Since finite-state grammars are a subset of the more general context-free grammar, we focus our discussion on context-free grammars only, although the parsing algorithm for finite-state grammars can be more efficient.

[1] The effort to prove natural languages are not context-free is summarized in Pullum and Gazdar [54].

Table 11.1 Chomsky hierarchy and the corresponding machine that accepts the language.

Types	Constraints	Automata
Phrase structure grammar	$\alpha \rightarrow \beta$. This is the most general grammar.	Turing machine
Context-sensitive grammar	A subset of the phrase structure grammar. $\|\alpha\| \leq \|\beta\|$, where |.| indicates the length of the string.	Linear bounded automata
Context-free grammar (CFG)	A subset of the context sensitive grammar. The production rule is $A \rightarrow \beta$, where A is a non-terminal. This production rule is shown to be equivalent to Chomsky normal form: $A \rightarrow w$ and $A \rightarrow BC$, where w is a terminal and B, C are non-terminals.	Push down automata
Regular grammar	A subset of the CFG. The production rule is expressed as: $A \rightarrow w$ and $A \rightarrow wB$.	Finite-state automata

As discussed in Section 11.1.2, a parsing algorithm offers a procedure that searches through various ways of combining grammatical rules to find a combination that generates a tree to illustrate the structure of the input sentence, which is similar to the search problem in speech recognition. The result of the parsing algorithm is a parse tree,[2] which can be regarded as a record of the CFG rules that account for the structure of the sentence. In other words, if we parse the sentence, working either top-down from S or bottom-up from each word, we automatically derive something that is similar to the tree representation, as illustrated in Figure 11.1.

A push-down automaton is also called a *recursive transition network* (RTN), which is an alternative formalism to describe context-free grammars. A transition network consists of nodes and labeled arcs. One of the nodes is specified as the initial state S. Starting at the initial state, we traverse an arc if the current word in the sentence is in the category on the arc. If the arc is followed, the current word is updated to the next word. A phrase can be parsed if there is a path from the starting node to a *pop* arc that indicates a complete parse for all the words in the phrase. Simple transition networks without recursion are often called *finite-state machines* (FSM). Finite-state machines are equivalent in expressive power to regular grammars and, thus, are not powerful enough to describe all languages that can be described by CFGs. Chapter 12 has a more detailed discussion on RTNs and FSMs used in speech recognition.

[2] The result can be more than one parse tree since natural language sentences are often ambiguous. In practice, a parsing algorithm should not only consider all the possible parse trees but also provide a ranking among them, as discussed in Chapter 17.

11.1.2. Chart Parsing for Context-Free Grammars

Since Chomsky introduced the notion of context-free grammars in the 1950s, a vast literature has arisen on the parsing algorithms. Most parsing algorithms were developed in computer science to analyze programming languages that are not ambiguous in the way that spoken language is [1, 32]. We discuss only the most relevant materials that are fundamental to building spoken language systems, namely the chart parser for the context-free grammar. This algorithm has been widely used in state-of-the-art spoken language understanding systems.

11.1.2.1. Top Down or Bottom Up?

Parsing is a special case of the search problem generally encountered in speech recognition. A parsing algorithm offers a procedure that searches through various ways of combining grammatical rules to find a combination that generates a tree to describe the structure of the input sentence, as illustrated in Figure 11.1. The search procedure can start from the root of the tree with the *S* symbol, attempting to rewrite it into a sequence of terminal symbols that matches the words in the input sentence, which is based on *goal-directed search*. Alternatively, the search procedure can start from the words in the input sentence and identify a word sequence that matches some non-terminal symbol. The bottom-up procedure can be repeated with partially parsed symbols until the root of the tree or the start symbol *S* is identified. This *data-directed search* has been widely used in practical SLU systems.

A top-down approach starts with the *S* symbol, then searches through different ways to rewrite the symbols until the input sentence is generated, or until all possibilities have been examined. A grammar is said to accept a sentence if there is a sequence of rules that allow us to rewrite the start symbol into the sentence. For the grammar in Figure 11.1, a sequence of rewrite rules can be illustrated as follows:

```
S
→ NP VP (rewriting S using S→NP)
→NAME VP (rewriting NP using NP→NAME)
→Mary VP (rewriting NAME using NAME→Mary)
...
→Mary loves that person (rewriting N using N→person)
```

Alternatively, we can take a bottom-up approach to start with the words in the input sentence and use the rewrite rules backward to reduce the sequence of symbols until it becomes S. The left-hand side of each rule is used to rewrite the symbol on the right-hand side as follows:

```
→NAME loves that person (rewriting Mary using NAME→Mary)
→NAME V that person (rewriting loves using V→loves)
...
→NP VP
→S (rewriting NP using S→NP VP)
```

A parsing algorithm must systematically explore every possible state that represents the intermediate node in the parsing tree. If a mistake occurs early on in choosing the rule that rewrites *S*, the intermediate parser results can be quite wasteful if the number of rules becomes large.

The main difference between top-down and bottom-up parsers is the way the grammar rules are used. For example, consider the rule *NP→ADJ NP1*. In a top-down approach, the rule is used to identify an *NP* by looking for the sequence *ADJ NP1*. Top-down parsing can be very predictive. A phrase or a word may be ambiguous in isolation. The top-down approach may prevent some ungrammatical combinations from consideration. It never wastes time exploring trees that cannot result in an *S*. On the other hand, it may predict many different constituents that do not have a match to the input sentence and rebuild large constituents again and again. For example, when the grammar is *left-recursive* (i.e., it contains a non-terminal category that has a derivation that includes itself anywhere along its leftmost branch), the top-down approach can lead a top-down, depth-first left-to-right parser to recursively expand the same non-terminal over again in exactly the same way. This causes an infinite expansion of trees. In contrast, a bottom-up parser takes a sequence *ADJ NP1* and identifies it as an *NP* according to the rule. The basic operation in bottom-up parsing is to take a sequence of symbols and match it to the right-hand side of the rules. It checks the input only once, and only builds each constituent exactly once. However, it may build up trees that have no hope of leading to *S* since it never suggests trees that are not at least locally grounded in the actual input. Since bottom-up parsing is similar to top-down parsing in terms of overall performance and is particularly suitable for robust spoken language processing as described in Chapter 17, we use the bottom-up method as our example to understand the key concept in the next section.

11.1.2.2. Bottom-Up Chart Parsing

As a standard search procedure, the state of the search consists of a symbol list, starting with the words in the sentence. Successor states can be generated by exploring all possible ways to replace a sequence of symbols that matches the right-hand side of a grammar rule with its left-hand side symbol. A simple-minded solution enumerates all the possible matches, leading to prohibitively expensive computational complexity. To avoid this problem, it is necessary to store partially parsed results of the matching, thereby eliminating duplicate work. This is the same technique that has been widely used in dynamic programming, as described in Chapter 8. Since chart parsing does not need to be from left to right, it is more efficient than the graph search algorithm discussed in Chapter 12, which can be used to parse the input sentence from left to right.

A data structure, called a *chart,* is used to allow the parser to store the partial results of the matching. The chart data structure maintains not only the records of all the constituents derived from the sentence so far in the parse tree, but also the records of rules that have matched partially but are still incomplete. These are called *active arcs*. Here, matches are always considered from the point of view of some *active constituents*, which represent the

subparts that the input sentence can be divided into according to the rewrite rules. Active constituents are stored in a data structure called an *agenda*. To find grammar rules that match a string involving the active constituent, we need to identify rules that start with the active constituent or rules that have already been started by earlier active constituents and require the current constituent to complete the rule or to extend the rule. The basic operation of a chart-based parser involves combining these partially matched records (active arcs) with a completed constituent to form either a new completed constituent or a new partially matched (but incomplete) constituent that is an extension of the original partially matched constituent. Just like the graph search algorithm, we can use either a depth-first or breadth-first search strategy, depending on how the agenda is implemented. If we use probabilities or other heuristics, we take the best-first strategy discussed in Chapter 12 to select constituents from the agenda. The chart-parser process is defined more precisely in Algorithm 11.1. It is possible to combine both top-down and bottom-up. The major difference is how the constituents are used.

ALGORITHM 11.1: *A BOTTOM-UP CHART PARSER*

Step1: Initialization: Define a list called chart to store active arcs, and a list called an agenda to store active constituents until they are added to the chart.
Step 2: Repeat: Repeat Step 2 to 7 until there is no input left.
Step 3: Push and pop the agenda: If the agenda is empty, look up the interpretations of the next word in the input and push them to the agenda. Pop a constituent *C* from the agenda. If *C* corresponds to position from w_i to w_j of the input sentence, we denote it *C[i,j]*.
Step 4: Add *C* to the chart: Insert *C[i,j]* into the chart.
Step 5: Add key-marked active arcs to the chart: For each rule in the grammar of the form $X \rightarrow C\ Y$, add to the chart an active arc (partially matched constituent) of the form $X[i,j] \rightarrow °CY$, where ° denotes the critical position called the key that indicates that everything before ° has been seen, but things after ° are yet to be matched (incomplete constituent).
Step 6: Move ° forward: For any active arc of the form $X[1,j] \rightarrow Y...°C...Z$ (everything before w_i) in the chart, add a new active arc of the form $X[1,j] \rightarrow Y...C°...Z$ to the chart.
Step 7: Add new constituents to the agenda: For any active arc of the form $X[1,l] \rightarrow Y...°C$, add a new constituent of type *X[1,j]* to the agenda.
Step 8: Exit: If *S*[1,*n*] is in the chart, where *n* is the length of the input sentence, we can exit successfully unless we want to find all possible interpretations of the sentence. The chart may contain many *S* structures covering the entire set of positions.

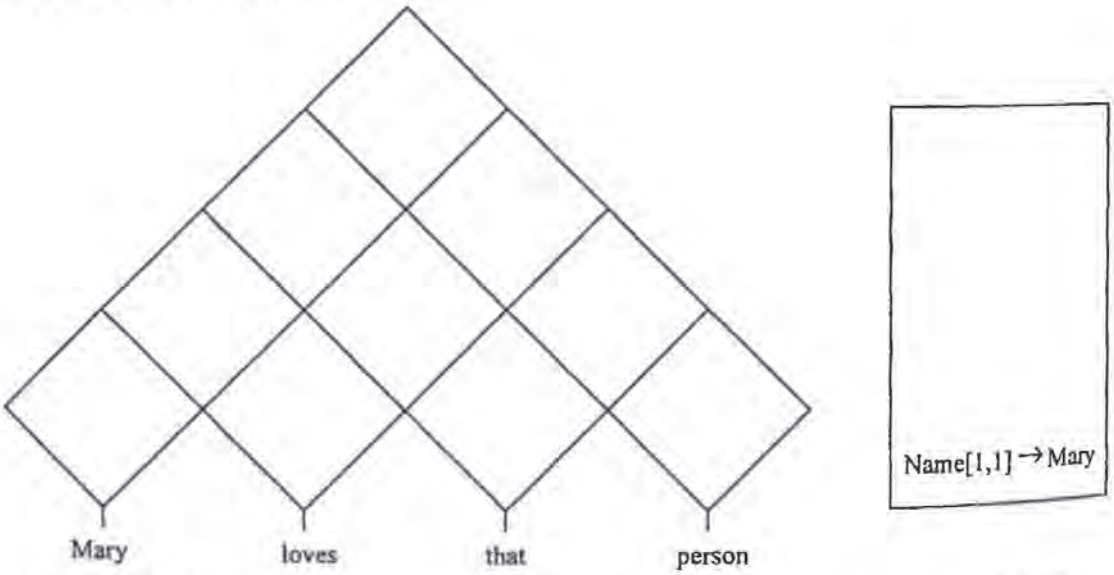
Let us look at an example to see how the chart parser parses the sentence *Mary loves that person* using the grammar specified in Figure 11.1. We first create the chart and agenda data structure as illustrated in Figure 11.2 (a), in which the leaves of the tree-like chart data structure corresponds to the position of each input word. The parent of each block in the chart covers from the position of the left child's corresponding starting word position to the right child's corresponding ending word position. Thus, the root block in the chart covers the whole sentence from the first word *Mary* to the last word *person*. The chart parser scans

through the input words to match against possible rewrite rules in the grammar. For the first word, the rule *Name→Mary* can be matched, so it is added to the agenda according to Step 3 in Algorithm 11.1. In Step 4, *Name→Mary* is added to the chart from the agenda. After the word Mary is processed, we have *Name→Mary, NP→Name*, and *S→NP°VP* in the chart, as illustrated in Figure 11.2 (b). *NP°VP* in the chart indicates that ° has reached the point at which everything before ° has been matched (in this case *Mary* matched *NP*) but everything after ° is yet to be parsed. The completed parsed chart is illustrated in Figure 11.2 (c).

A parser may assign one or more parsed structures to the sentence in the language it defines. If any sentence is assigned more than one such structure, the grammar is said to be ambiguous. Spoken language is, of course, ambiguous by nature.[3] For example, we can have a sentence like *Mary sold the student bags*. It is unclear whether *student* should be the modifier for *bags* or whether it means that *Mary* sold the *bags* to *the student*.

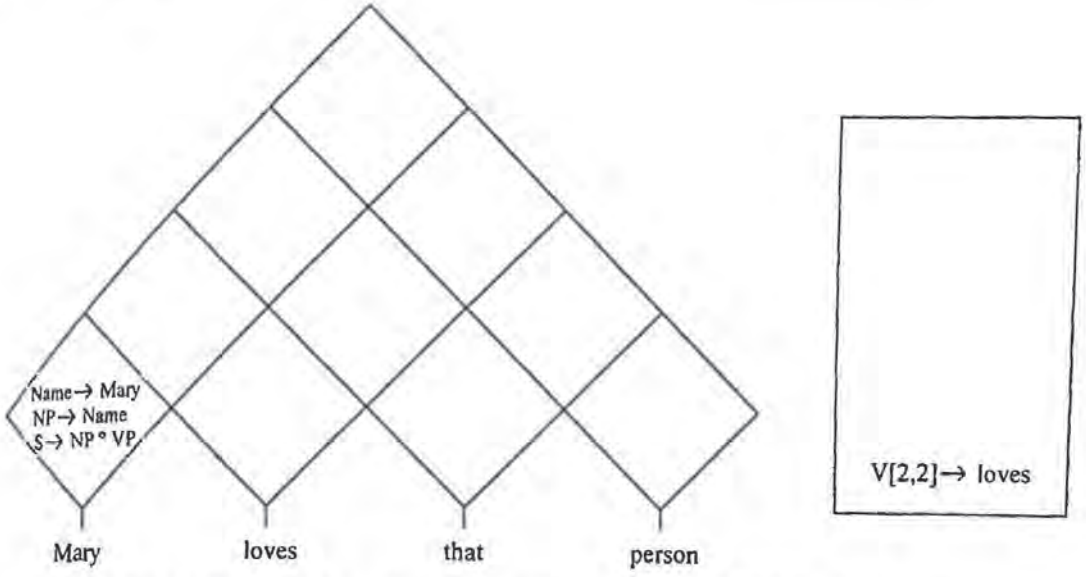
Chart parsers can be fairly efficient simply because the same constituent is never constructed more than once. In the worst case, the chart parser builds every possible constituent between every possible pair of positions, leading to the worst-case computational complexity of $O(n^3)$, where n is the length of the input sentence. This is still far more efficient than a straightforward brute-force search.

In many practical tasks, we need only a partial parse or shallow parse of the input sentence. You can use cascades of finite-state automata instead of CFGs. Relying on simple finite-state automata rather than full parsing makes such systems more efficient, although finite-state systems cannot model certain kinds of recursive rules, so that efficiency is traded for a certain lack of coverage.

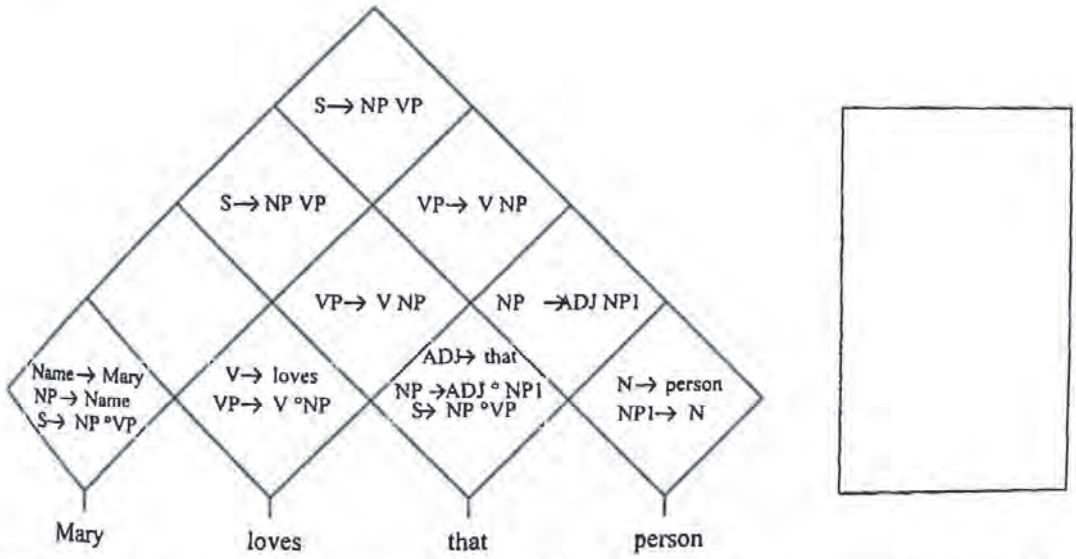


(a) The chart is illustrated on the left, and the agenda is on the right. The agenda now has one rule in it according to Step 3, since the agenda is empty.

[3] The same parse tree can also mean multiple things, so a parse tree itself does not define meaning. "*Mary loves that person*" could be sarcastic and mean something different.



(b) After Mary, the chart now has rules *Name→Mary, NP→Name*, and *S→NP°VP*.



(c) The chart after the whole sentence is parsed. S→ NP VP covers the whole sentence, indicating that the sentence is parsed successfully by the grammar.

Figure 11.2 An example of a chart parser with the grammar illustrated in Figure 11.1. Parts (a) and (b) show the initial chart and agenda to parse the first word; part (c) shows the chart after the sentence is completely parsed.

11.2. STOCHASTIC LANGUAGE MODELS

Stochastic language models (SLM) take a probabilistic viewpoint of language modeling. We need to accurately estimate the probability $P(\mathbf{W})$ for a given word sequence $\mathbf{W} = w_1 w_2 ... w_n$. In the formal language theory discussed in Section 11.1, $P(\mathbf{W})$ can be regarded as 1 or 0 if the word sequence is accepted or rejected, respectively, by the grammar. This may be inappropriate for spoken language systems, since the grammar itself is unlikely to have a complete coverage, not to mention that spoken language is often ungrammatical in real conversational applications.

The key goal of SLM is to provide adequate probabilistic information so that likely word sequences should have a higher probability. This not only makes speech recognition more accurate but also helps to dramatically constrain the search space for speech recognition (see Chapters 12 and 13). Notice that SLM can have a wide coverage on all the possible word sequences, since probabilities are used to differentiate different word sequences. The most widely used SLM is the so call n-gram model discussed in this chapter. In fact, the CFG can be augmented as the bridge between the n-gram and the formal grammar if we can incorporate probabilities into the production rules, as discussed in the next section.

11.2.1. Probabilistic Context-Free Grammars

The CFG can be augmented with probability for each production rule. The advantages of probabilistic CFGs (PCFGs) lie in their ability to more accurately capture the embedded usage structure of spoken language to minimize syntactic ambiguity. The use of probability becomes increasingly important to discriminate many competing choices when the number of rules is large.

In the PCFG, we have to address the parallel problems we discussed for HMMs in Chapter 8. The *recognition problem* is concerned with the computation of the probability of the start symbol S generating the word sequence $\mathbf{W} = w_1, w_2, ... w_T$, given the grammar G:

$$P(S \Rightarrow \mathbf{W}|G) \tag{11.1}$$

where $\Rightarrow$ denotes a derivation sequence consisting of one or more steps. This is equivalent to the chart parser augmented with probabilities, as discussed in Section 11.1.2.2.

The *training problem* is concerned with determining a set of rules G based on the training corpus and estimating the probability of each rule. If the set of rules is fixed, the simplest approach to deriving these probabilities is to count the number of times each rule is used in a corpus containing parsed sentences. We denote the probability of a rule $A \rightarrow \alpha$ by $P(A \rightarrow \alpha|G)$. For instance, if there are m rules for left-hand side non-terminal node A: $A \rightarrow \alpha_1, A \rightarrow \alpha_2, ... A \rightarrow \alpha_m$, we can estimate the probability of these rules as follows:

$$P(A \rightarrow \alpha_j | G) = C(A \rightarrow \alpha_j) / \sum_{i=1}^{m} C(A \rightarrow \alpha_i) \tag{11.2}$$

where $C(.)$ denotes the number of times each rule is used.

When you have hand-annotated corpora, you can use the maximum likelihood estimation as illustrated by Eq. (11.2) to derive the probabilities. When you don't have hand-annotated corpora, you can extend the EM algorithm (see Chapter 4) to derive these probabilities. The algorithm is also known as the *inside-outside* algorithm. As we discussed in Chapter 8, you can develop algorithms similar to the Viterbi algorithm to find the most likely parse tree that could have generated the sequence of words $P(\mathbf{W})$ after these probabilities are estimated.

We can make certain independence assumptions about rule usage. Namely, we assume that the probability of a constituent being derived by a rule is independent of how the constituent is used as a subconstituent. For instance, we assume that the probabilities of *NP* rules are the same no matter whether the *NP* is used for the subject or the object of a verb, although the assumptions are not valid in many cases. More specifically, let the word sequence $\mathbf{W}=w_1 w_2 \ldots w_T$ be generated by a PCFG G, with rules in Chomsky normal form as discussed in Section 11.1.1:

$$A_i \rightarrow A_m A_n \text{ and } A_i \rightarrow w_l \tag{11.3}$$

where A_m and A_n are two possible non-terminals that expand A_i at different locations. The probability for these rules must satisfy the following constraint:

$$\sum_{m,n} P(A_i \rightarrow A_m A_n \mid G) + \sum_{l} P(A_i \rightarrow w_l \mid G) = 1, \quad \text{for all } i \tag{11.4}$$

Equation (11.4) simply means that all non-terminals can generate either pairs of non-terminal symbols or a single terminal symbol, and all these production rules should satisfy the probability constraint. Analogous to the HMM forward and backward probabilities discussed in Chapter 8, we can define the inside and outside probabilities to facilitate the estimation of these probabilities from the training data.

A non-terminal symbol A_i can generate a sequence of words $w_j w_{j+1} \ldots w_k$; we define the probability of $Inside(j, A_i, k) = P(A_i \Rightarrow w_j w_{j+1} \ldots w_k \mid G)$ as the *inside constituent probability*, since it assigns a probability to the word sequence inside the constituent. The inside probability can be computed recursively. When only one word is emitted, the transition rule of the form $A_i \rightarrow w_m$ applies. When there is more than one word, rules of the form $A_i \rightarrow A_m A_n$ must apply. The inside probability of $inside(j, A_i, k)$ can be expressed recursively as follows:

$$\begin{aligned} inside(j, A_i, k) &= P(A_i \Rightarrow w_j w_{j+1} \ldots w_k) \\ &= \sum_{n,m} \sum_{l=j}^{k-1} P(A_i \rightarrow A_m A_n) P(A_m \Rightarrow w_j \ldots w_l) P(A_n \Rightarrow w_{l+1} \ldots w_k) \\ &= \sum_{n,m} \sum_{l=j}^{k-1} P(A_l \rightarrow A_m A_n) inside(j, A_m, l) inside(l+1, A_n, k) \end{aligned} \tag{11.5}$$

The inside probability is the sum of the probabilities of all derivations for the section over the span of j to k. One possible derivation of the form can be drawn as a parse tree shown in Figure 11.3.

Another useful probability is the *outside* probability for a non-terminal node A_i covering w_s to w_t, in which they can be derived from the start symbol S, as illustrated in Figure 11.4, together with the rest of the words in the sentence:

$$outside(s, A_i, t) = P(S \Rightarrow w_1 \ldots w_{s-1}\ A_i\ w_{t+1} \ldots w_T) \tag{11.6}$$

After the inside probabilities are computed bottom-up, we can compute the outside probabilities top-down. For each non-terminal symbol A_j, there are one of two possible configurations $A_m \rightarrow A_n A_j$ or $A_m \rightarrow A_j A_n$ as illustrated in Figure 11.5. Thus, we need to consider all the possible derivations of these two forms as follows:

$$\begin{aligned} outside(s, A_j, t) &= P(S \Rightarrow w_1 \ldots w_{s-1}\ A_j\ w_{t+1} \ldots w_T) \\ &= \sum_{m,n} \left\{ \begin{array}{l} \sum_{l=1}^{s-1} P(A_m \rightarrow A_n A_j) P(A_n \Rightarrow w_l \ldots w_{s-1}) P(S \Rightarrow w_1 \ldots w_{l-1}\ A_m\ w_{t+1} \ldots w_T) + \\ + \sum_{l=t+1}^{T} P(A_m \rightarrow A_j A_n) P(A_n \Rightarrow w_{t+1} \ldots w_l) P(S \Rightarrow w_1 \ldots w_{s-1}\ A_m\ w_{t+1} \ldots w_T) \end{array} \right\} \\ &= \sum_{m,n} \left\{ \begin{array}{l} \sum_{l=1}^{s-1} P(A_m \rightarrow A_n A_j) inside(l, A_n, s-1) outside(l, A_m, t) + \\ + \sum_{l=t+1}^{T} P(A_m \rightarrow A_j A_n) inside(t+1, A_n, l) outside(s, A_m, l) \end{array} \right\} \end{aligned} \tag{11.7}$$

The inside and outside probabilities are used to compute the sentence probability as follows:

$$P(S \Rightarrow w_1 \ldots w_T) = \sum_j inside(s, A_j, t) outside(s, A_j, t) \qquad \text{for any } s \le t \tag{11.8}$$

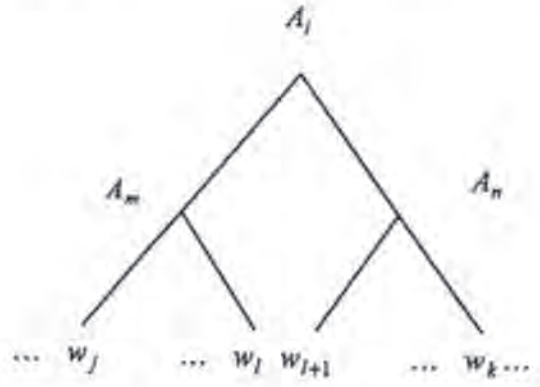


Figure 11.3 Inside probability is computed recursively as sum of all the derivations.

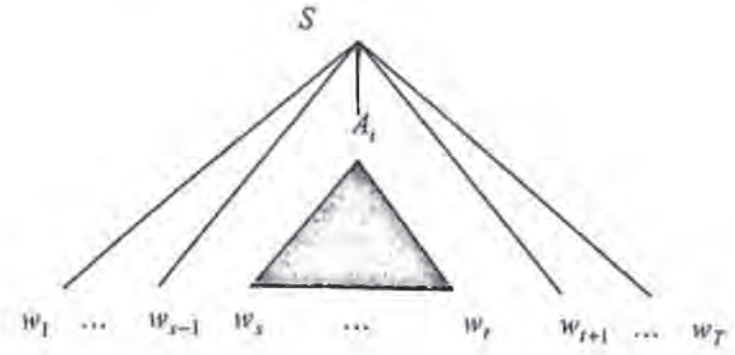


Figure 11.4 Definition of the outside probability.

Since $outside(1, A_i, T)$ is equal to 1 for the starting symbol only, the probability for the whole sentence can be conveniently computed using the inside probability alone as

$$P(S \Rightarrow \mathbf{W}|G) = inside(1, S, T) \tag{11.9}$$

We are interested in the probability that a particular rule, $A_i \rightarrow A_m A_n$ is used to cover a span $w_s \ldots w_t$, given the sentence and the grammar:

$$\xi(i,m,n,s,t) = P(A_i \Rightarrow w_s \ldots w_t, A_i \rightarrow A_m A_n \mid S \Rightarrow \mathbf{W}, G)$$
$$= \frac{1}{P(S \Rightarrow \mathbf{W} \mid G)} \sum_{k=s}^{t-1} P(A_i \rightarrow A_m A_n \mid G) inside(s, A_m, k) inside(k+1, A_n, t) outside(s, A_i, t) \tag{11.10}$$

These conditional probabilities form the basis of the inside-outside algorithm, which is similar to the forward-backward algorithm discussed in Chapter 8. We can start with some initial probability estimates. For each sentence of training data, we determine the inside and outside probabilities in order to compute, for each production rule, how likely it is that the production rule is used as part of the derivation of that sentence. This gives us the number of counts for each production rule in each sentence. Summing these counts across sentences gives us an estimate of the total number of times each production rule is used to produce the

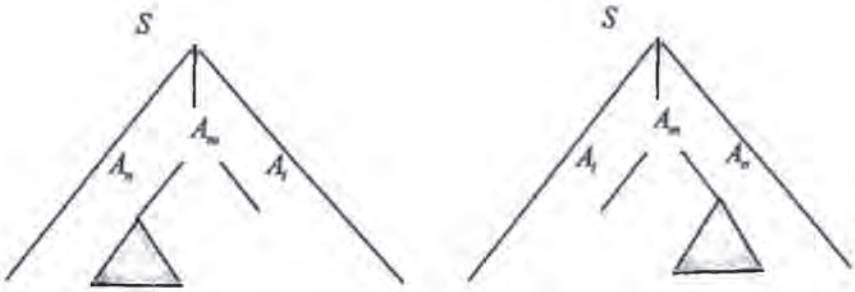


Figure 11.5 Two possible configurations for a non-terminal node A_i.

sentences in the training corpus. Dividing by the total counts of productions used for each non-terminal gives us a new estimate of the probability of the production in the MLE framework. For example, we have:

$$P(A_i \rightarrow A_m A_n | G) = \frac{\sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \xi(i,m,n,s,t)}{\sum_{m,n} \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \xi(i,m,n,s,t)} \tag{11.11}$$

In a similar manner, we can estimate $P(A_i \rightarrow w_m | G)$. It is also possible to let the inside-outside algorithm formulate all the possible grammar production rules so that we can select rules with sufficient probability values. If there is no constraint, we may have too many *greedy symbols* that serve as possible non-terminals. In addition, the algorithm is guaranteed only to find a local maximum. It is often necessary to use prior knowledge about the task and the grammar to impose strong constraints to avoid these two problems. The chart parser discussed in Section 11.1.2 can be modified to accommodate PCFGs [29, 45].

One problem with the PCFG is that it assumes that the expansion of any one non-terminal is independent of the expansion of other non-terminals. Thus each PCFG rule probability is multiplied together without considering the location of the node in the parse tree. This is against our intuition since there is a strong tendency toward the context-dependent expansion. Another problem is its lack of sensitivity to words, although lexical information plays an important role in selecting the correct parsing of an ambiguous prepositional phrase attachment. In the PCFG, lexical information can only be represented via the probability of pre-terminal nodes, such as verb or noun, to be expanded lexically. You can add lexical dependencies to PCFGs and make PCFG probabilities more sensitive to surrounding syntactic structure [6, 11, 19, 31, 45].

11.2.2. *N*-gram Language Models

As covered earlier, a language model can be formulated as a probability distribution $P(\mathbf{W})$ over word strings $\mathbf{W}$ that reflects how frequently a string $\mathbf{W}$ occurs as a sentence. For example, for a language model describing spoken language, we might have *P(hi)* = 0.01, since perhaps one out of every hundred sentences a person speaks is *hi*. On the other hand, we would have *P(lid gallops Changsha pop)* = 0, since it is extremely unlikely anyone would utter such a strange string.

$P(\mathbf{W})$ can be decomposed as

$$\begin{aligned} P(\mathbf{W}) &= P(w_1, w_2, \ldots, w_n) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\cdots P(w_n|w_1, w_2, \ldots, w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1, w_2, \ldots, w_{i-1}) \end{aligned} \tag{11.12}$$

where $P(w_i|w_1, w_2, \ldots, w_{i-1})$ is the probability that w_i will follow, given that the word sequence $w_1, w_2, \ldots, w_{i-1}$ was presented previously. In Eq. (11.12), the choice of w_i thus depends on the entire past history of the input. For a vocabulary of size v there are v^{i-1} different histories and so, to specify $P(w_i|w_1, w_2, \ldots, w_{i-1})$ completely, v^i values would have to be estimated. In reality, the probabilities $P(w_i|w_1, w_2, \ldots, w_{i-1})$ are impossible to estimate for even moderate values of i, since most histories $w_1, w_2, \ldots, w_{i-1}$ are unique or have occurred only a few times. A practical solution to the above problems is to assume that $P(w_i|w_1, w_2, \ldots, w_{i-1})$ depends only on some equivalence classes. The equivalence class can be simply based on the several previous words $w_{i-N+1}, w_{i-N+2}, \ldots, w_{i-1}$. This leads to an *n-gram* language model. If the word depends on the previous two words, we have a *trigram*: $P(w_i|w_{i-2}, w_{i-1})$. Similarly, we can have *unigram*: $P(w_i)$, or *bigram*: $P(w_i|w_{i-1})$ language models. The trigram is particularly powerful, as most words have a strong dependence on the previous two words, and it can be estimated reasonably well with an attainable corpus.

In bigram models, we make the approximation that the probability of a word depends only on the identity of the immediately preceding word. To make $P(w_i|w_{i-1})$ meaningful for $i = 1$, we pad the *beginning of the sentence* with a distinguished token <s>; that is, we pretend $w_0 = $ <s>. In addition, to make the sum of the probabilities of all strings equal 1, it is necessary to place a distinguished token </s> at the *end of the sentence*. For example, to calculate *P(Mary loves that person)* we would take

$$P(\textit{Mary loves that person}) =$$
$$P(\textit{Mary}|\langle s\rangle)P(\textit{loves}|\textit{Mary})P(\textit{that}|\textit{loves})P(\textit{person}|\textit{that})P(\langle/s\rangle|\textit{person})$$

To estimate $P(w_i|w_{i-1})$, the frequency with which the word w_i occurs given that the last word is w_{i-1}, we simply count how often the sequence (w_{i-1}, w_i) occurs in some text and normalize the count by the number of times w_{i-1} occurs.

In general, for a trigram model, the probability of a word depends on the two preceding words. The trigram can be estimated by observing the frequencies or counts of the word pair $C(w_{i-2}, w_{i-1})$ and triplet $C(w_{i-2}, w_{i-1}, w_i)$ as follows:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \tag{11.13}$$

The text available for building a model is called a training corpus. For *n*-gram models, the amount of training data used is typically many millions of words. The estimate of Eq. (11.13) is based on the maximum likelihood principle, because this assignment of probabilities yields the trigram model that assigns the highest probability to the training data of all possible trigram models.

We sometimes refer to the value n of an n-gram model as its order. This terminology comes from the area of Markov models, of which n-gram models are an instance. In particular, an n-gram model can be interpreted as a Markov model of order n-1.

Consider a small example. Let our training data S be comprised of the three sentences *John read her book. I read a different book. John read a book by Mulan*. and let us calculate *P(John read a book)* for the maximum likelihood bigram model. We have

$$P(John|<s>)=\frac{C(<s>,John)}{C(<s>)}=\frac{2}{3}$$

$$P(read|John)=\frac{C(John,read)}{C(John)}=\frac{2}{2}$$

$$P(a|read)=\frac{C(read,a)}{C(read)}=\frac{2}{3}$$

$$P(book|a)=\frac{C(a,book)}{C(a)}=\frac{1}{2}$$

$$P(</s>|book)=\frac{C(book,</s>)}{C(book)}=\frac{2}{3}$$

These trigram probabilities help us estimate the probability for the sentence as:

$$\begin{aligned}&P(John\ read\ a\ book)\\&=P(John|<s>)P(read|John)P(a|read)P(book|a)P(</s>|book)\\&\approx 0.148\end{aligned}\qquad(11.14)$$

If these three sentences are all the data we have available to use in training our language model, the model is unlikely to generalize well to new sentences. For example, the sentence "*Mulan read her book*" should have a reasonable probability, but the trigram will give it a zero probability simply because we do not have a reliable estimate for *P(read|Mulan)*.

Unlike linguistics, grammaticality is not a strong constraint in the *n*-gram language model. Even though the string is ungrammatical, we may still assign it a high probability if *n* is small.

11.3. Complexity Measure of Language Models

Language can be thought of as an information source whose outputs are words w_i belonging to the vocabulary of the language. The most common metric for evaluating a language model is the word recognition error rate, which requires the participation of a speech recognition system. Alternatively, we can measure the probability that the language model assigns to test word strings without involving speech recognition systems. This is the derivative measure of cross-entropy known as test-set *perplexity*.

The measure of cross-entropy is discussed in Chapter 3. Given a language model that assigns probability $P(\mathbf{W})$ to a word sequence $\mathbf{W}$, we can derive a compression algorithm that encodes the text $\mathbf{W}$ using $-\log_2 P(\mathbf{W})$ bits. The cross-entropy $H(\mathbf{W})$ of a model

$P(w_i|w_{i-n+1}\cdots w_{i-1})$ on data $\mathbf{W}$, with a sufficiently long word sequence, can be simply approximated as

$$H(\mathbf{W}) = -\frac{1}{N_w}\log_2 P(\mathbf{W}) \tag{11.15}$$

where N_w is the length of the text $\mathbf{W}$ measured in words.

The perplexity $PP(\mathbf{W})$ of a language model $P(\mathbf{W})$ is defined as the reciprocal of the (geometric) average probability assigned by the model to each word in the test set $\mathbf{W}$. This is a measure, related to cross-entropy, known as test-set perplexity:

$$PP(\mathbf{W}) = 2^{H(\mathbf{W})} \tag{11.16}$$

The perplexity can be roughly interpreted as the geometric mean of the branching factor of the text when presented to the language model. The perplexity defined in Eq. (11.16) has two key parameters: a language model and a word sequence. The test-set[4] perplexity evaluates the generalization capability of the language model. The training-set perplexity measures how the language model fits the training data, like the likelihood. It is generally true that lower perplexity correlates with better recognition performance. This is because the perplexity is essentially a statistically weighted word branching measure on the test set. The higher the perplexity, the more branches the speech recognizer needs to consider statistically.

While the perplexity [Eqs. (11.16) and (11.15)] is easy to calculate for the n-gram [Eq. (11.12)], it is slightly more complicated to compute for a probabilistic CFG. We can first parse the word sequence and use Eq. (11.9) to compute $P(\mathbf{W})$ for the test-set perplexity. The perplexity can also be applied to nonstochastic models such as CFGs. We can assume they have a uniform distribution in computing $P(\mathbf{W})$.

A language with higher perplexity means that the number of words branching from a previous word is larger on average. In this sense, perplexity is an indication of the complexity of the language if we have an accurate estimate of $P(\mathbf{W})$. For a given language, the difference between the perplexity of a language model and the true perplexity of the language is an indication of the quality of the model. The perplexity of a particular language model can change dramatically in terms of the vocabulary size, the number of states of grammar rules, and the estimated probabilities. A language model with perplexity X has roughly the same difficulty as another language model in which every word can be followed by X different words with equal probabilities. Therefore, in the task of continuous digit recognition, the perplexity is 10. Clearly, lower perplexity will generally have less confusion in recognition. Typical perplexities yielded by n-gram models on English text range from about 50 to almost 1000 (corresponding to cross-entropies from about 6 to 10 bits/word), depending on the type of text. In the task of 5,000-word continuous speech recognition for the *Wall Street Journal*, the test-set perplexities of the trigram grammar and the bigram grammar are re-

[4] We often distinguish between the word sequence from the unseen test data and that from the training data to derive the language model.

ported to be about 128 and 176 respectively.[5] In the tasks of 2000-word conversational Air Travel Information System (ATIS), the test-set perplexity of the word trigram model is typically less than 20.

Since perplexity does not take into account acoustic confusability, we eventually have to measure speech recognition accuracy. For example, if the vocabulary of a speech recognizer contains the E-set of English alphabet: *B, C, D, E, G, P*,and *T*, we can define a CFG that has a low perplexity value of 7. Such a low perplexity does not guarantee we will have good recognition performance, because of the intrinsic acoustic confusability of the E-set.

11.4. *N*-GRAM SMOOTHING

One of the key problems in *n*-gram modeling is the inherent data sparseness of real training data. If the training corpus is not large enough, many actually possible word successions may not be well observed, leading to many extremely small probabilities. For example, with several-million-word collections of English text, more than 50% of trigrams occur only once, and more than 80% of trigrams occur less than five times. Smoothing is critical to make estimated probabilities robust for unseen data. If we consider the sentence *Mulan read a book* in the example we discussed in Section 11.2.2, we have:

$$P(read|\, Mulan) = \frac{C(Mulan, read)}{\sum_{w} C(Mulan, w)} = \frac{0}{1}$$

giving us $P(Mulan\ read\ a\ book) = 0$.

Obviously, this is an underestimate for the probability of "*Mulan read a book*" since there is *some* probability that the sentence occurs in some test set. To show why it is important to give this probability a nonzero value, we turn to the primary application for language models, speech recognition. In speech recognition, if $P(\mathbf{W})$ is zero, the string **W** will never be considered as a possible transcription, regardless of how unambiguous the acoustic signal is. Thus, whenever a string W such that $P(\mathbf{W}) = 0$ occurs during a speech recognition task, an error will be made. Assigning all strings a nonzero probability helps prevent errors in speech recognition. This is the core issue of smoothing. Smoothing techniques adjust the maximum likelihood estimate of probabilities to produce more robust probabilities for unseen data, although the likelihood for the training data may be hurt slightly.

The name smoothing comes from the fact that these techniques tend to make distributions flatter, by adjusting low probabilities such as zero probabilities upward, and high probabilities downward. Not only do smoothing methods generally prevent zero probabilities,

[5] Some experimental results show that the test-set perplexities for different languages are comparable. For example, French, English, Italian and German have a bigram test-set perplexity in the range of 95 to 133 for newspaper corpora. Italian has a much higher perplexity reduction (a factor of 2) from bigram to trigram because of the high number of function words. The trigram perplexity of Italian is among the lowest in these languages [34].

but they also attempt to improve the accuracy of the model as a whole. Whenever a probability is estimated from few counts, smoothing has the potential to significantly improve the estimation so that it has better generalization capability.

To give an example, one simple smoothing technique is to pretend each bigram occurs once more than it actually does, yielding

$$P(w_i|w_{i-1}) = \frac{1+C(w_{i-1},w_i)}{\sum_{w_i}(1+C(w_{i-1},w_i))} = \frac{1+C(w_{i-1},w_i)}{V+\sum_{w_i}C(w_{i-1},w_i)} \tag{11.17}$$

where V is the size of the vocabulary. In practice, vocabularies are typically fixed to be tens of thousands of words or less. All words not in the vocabulary are mapped to a single word, usually called the *unknown word*. Let us reconsider the previous example using this new distribution, and let us take our vocabulary V to be the set of all words occurring in the training data S, so that we have $V = 11$ (with both <s> and </s>).

For the sentence *John read a book*, we now have

$$\begin{aligned} &P(John\ read\ a\ book) \\ &= P(John|</s>)P(read|John)P(a|read)P(book|a)P(</s>|book) \\ &\approx 0.00035 \end{aligned} \tag{11.18}$$

In other words, we estimate that the sentence *John read a book* occurs about once every three thousand sentences. This is more reasonable than the maximum likelihood estimate of 0.148 of Eq. (11.14). For the sentence *Mulan read a book*, we have

$$\begin{aligned} &P(Mulan\ read\ a\ book) \\ &= P(Mulan|</s>)P(read|Mulan)P(a|read)P(book|a)P(</s>|book) \\ &\approx 0.000084 \end{aligned} \tag{11.19}$$

Again, this is more reasonable than the zero probability assigned by the maximum likelihood model. In general, most existing smoothing algorithms can be described with the following equation:

$$\begin{aligned} &P_{smooth}(w_i \mid w_{i-n+1}...w_{i-1}) \\ &= \begin{cases} \alpha(w_i \mid w_{i-n+1}...w_{i-1}) & \text{if } C(w_{i-n+1}...w_i)>0 \\ \gamma(w_{i-n+1}...w_{i-1})P_{smooth}(w_i \mid w_{i-n+2}...w_{i-1}) & \text{if } C(w_{i-n+1}...w_i)=0 \end{cases} \end{aligned} \tag{11.20}$$

That is, if an n-gram has a nonzero count we use the distribution $\alpha(w_i|w_{i-n+1}...w_{i-1})$. Otherwise, we backoff to the lower-order n-gram distribution $P_{smooth}(w_i|w_{i-n+2}...w_{i-1})$, where the scaling factor $\gamma(w_{i-n+1}...w_{i-1})$ is chosen to make the conditional distribution sum to one. We refer to algorithms that fall directly in this framework as *backoff models*.

Several other smoothing algorithms are expressed as the linear interpolation of higher- and lower-order n-gram models as:

$$P_{smooth}(w_i \mid w_{i-n+1} \ldots w_{i-1}) = \lambda P_{ML}(w_i \mid w_{i-n+1} \ldots w_{i-1}) + (1-\lambda) P_{smooth}(w_i \mid w_{i-n+2} \ldots w_{i-1}) \tag{11.21}$$

where λ is the interpolation weight that depends on $w_{i-n+1} \ldots w_{i-1}$. We refer to models of this form as interpolated models.

The key difference between backoff and interpolated models is that for the probability of n-grams with nonzero counts, interpolated models use information from lower-order distributions while backoff models do not. In both backoff and interpolated models, lower-order distributions are used in determining the probability of n-grams with zero counts. Now, we discuss several backoff and interpolated smoothing methods. Performance comparison of these techniques in real speech recognition applications is discussed in Section 11.4.4.

11.4.1. Deleted Interpolation Smoothing

Consider the case of constructing a bigram model on training data where we have that *C(enliven you)* = 0 *and C(enliven thou)* = 0. Then, according to both additive smoothing of Eq. (11.17), we have *P(you|enliven) = P(thou|enliven)*. However, intuitively we should have *P(you|enliven) > P(thou|enliven)*, because the word *you* is much more common than the word *thou* in modern English. To capture this behavior, we can interpolate the bigram model with a unigram model. A unigram model conditions the probability of a word on no other words, and just reflects the frequency of that word in text. We can linearly interpolate a bigram model and a unigram model as follows:

$$P_I(w_i \mid w_{i-1}) = \lambda P(w_i \mid w_{i-1}) + (1-\lambda) P(w_i) \tag{11.22}$$

where $0 \le \lambda \le 1$. Because *P(you|enliven) = P(thou|enliven)*=0 while presumably *P(you) > P(thou)*, we will have that $P_I(you \mid enliven) > P_I(thou \mid enliven)$ as desired.

In general, it is useful to interpolate higher-order n-gram models with lower-order n-gram models, because when there is insufficient data to estimate a probability in the higher-order model, the lower-order model can often provide useful information. An elegant way of performing this interpolation is given as follows

$$P_I(w_i \mid w_{i-n+1} \cdots w_{i-1}) = \lambda_{w_{i-n+1} \ldots w_{i-1}} P(w_i \mid w_{i-n+1} \cdots w_{i-1}) + (1-\lambda_{w_{i-n+1} \ldots w_{i-1}}) P_I(w_i \mid w_{i-n+2} \cdots w_{i-1}) \tag{11.23}$$

That is, the nth-order smoothed model is defined *recursively* as a linear interpolation between the nth-order maximum likelihood model and the $(n-1)$th-order smoothed model. To end the recursion, we can take the smoothed first-order model to be the maximum likeli-

hood distribution (unigram), or we can take the smoothed zeroth-order model to be the uniform distribution. Given a fixed $P(w_i|w_{i-n+1}\cdots w_{i-1})$, it is possible to search efficiently for the interpolation parameters using the deleted interpolation method discussed in Chapter 9.

Notice that the optimal $\lambda_{w_{i-n+1}\cdots w_{i-1}}$ is different for different histories $w_{i-n+1}\cdots w_{i-1}$. For example, for a context we have seen thousands of times, a high λ will be suitable, since the higher-order distribution is very reliable; for a history that has occurred only once, a lower λ is appropriate. Training each parameter $\lambda_{w_{i-n+1}\cdots w_{i-1}}$ independently can be harmful; we need an enormous amount of data to train so many independent parameters accurately. One possibility is to divide the $\lambda_{w_{i-n+1}\cdots w_{i-1}}$ into a moderate number of partitions or buckets, constraining all $\lambda_{w_{i-n+1}\cdots w_{i-1}}$ in the same bucket to have the same value, thereby reducing the number of independent parameters to be estimated. Ideally, we should tie together those $\lambda_{w_{i-n+1}\cdots w_{i-1}}$ that we have a prior reason to believe should have similar values.

11.4.2. Backoff Smoothing

Backoff smoothing is attractive because it is easy to implement for practical speech recognition systems. The Katz backoff model is the canonical example we discuss in this section. It is based on the Good-Turing smoothing principle.

11.4.2.1. Good-Turing Estimates and Katz Smoothing

The Good-Turing estimate is a smoothing technique to deal with infrequent n-grams. It is not used by itself for n-gram smoothing, because it does not include the combination of higher-order models with lower-order models necessary for good performance. However, it is used as a tool in several smoothing techniques. The basic idea is to partition n-grams into groups depending on their frequency (i.e. how many time the n-grams appear in the training data) such that the parameter can be smoothed based on n-gram frequency.

The Good-Turing estimate states that for any n-gram that occurs r times, we should pretend that it occurs r^* times as follows:

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{11.24}$$

where n_r is the number of n-grams that occur exactly r times in the training data. To convert this count to a probability, we just normalize: for an n-gram a with r counts, we take

$$P(a) = \frac{r^*}{N} \tag{11.25}$$

where $N = \sum_{r=0}^{\infty} n_r r^*$. Notice that $N = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty}(r+1)n_{r+1} = \sum_{r=0}^{\infty} n_r r$, i.e., N is equal to the original number of counts in the distribution [28].

Katz smoothing extends the intuitions of the Good-Turing estimate by adding the combination of higher-order models with lower-order models [38]. Take the bigram as our example, Katz smoothing suggested using the Good-Turing estimate for nonzero counts as follows:

$$C^*(w_{i-1}w_i) = \begin{cases} d_r r & \text{if } r>0 \\ \alpha(w_{i-1})P(w_i) & \text{if } r=0 \end{cases} \tag{11.26}$$

where d_r is approximately equal to r^*/r. That is, all bigrams with a nonzero count r are *discounted* according to a discount ratio d_r, which implies that the counts subtracted from the nonzero counts are distributed among the zero-count bigrams according to the next lower-order distribution, e.g., the unigram model. The value $\alpha(w_{i-1})$ is chosen to equalize the total number of counts in the distribution, i.e., $\sum_{w_i} C^*(w_{i-1}w_i) = \sum_{w_i} C(w_{i-1}w_i)$. The appropriate value for $\alpha(w_{i-1})$ is computed so that the smoothed bigram satisfies the probability constraint:

$$\alpha(w_{i-1}) = \frac{1-\sum_{w_i:C(w_{i-1}w_i)>0} P^*(w_i|w_{i-1})}{\sum_{w_i:C(w_{i-1}w_i)=0} P(w_i)} = \frac{1-\sum_{w_i:C(w_{i-1}w_i)>0} P^*(w_i|w_{i-1})}{1-\sum_{w_i:C(w_{i-1}w_i)>0} P(w_i)} \tag{11.27}$$

To calculate $P^*(w_i|w_{i-1})$ from the corrected count, we just normalize:

$$P^*(w_i \mid w_{i-1}) = \frac{C^*(w_{i-1}w_i)}{\sum_{w_k} C^*(w_{i-1}w_k)} \tag{11.28}$$

In Katz implementation, the d_r are calculated as follows: large counts are taken to be reliable, so they are not discounted. In particular, Katz takes $d_r = 1$ for all $r > k$ for some k, say k in the range of 5 to 8. The discount ratios for the lower counts $r \le k$ are derived from the Good-Turing estimate applied to the global bigram distribution; that is, n_r in Eq. (11.24) denotes the total number of bigrams that occur exactly r times in the training data. These d_r are chosen such that

- the resulting discounts are proportional to the discounts predicted by the Good-Turing estimate, and
- the total number of counts discounted in the global bigram distribution is equal to the total number of counts that should be assigned to bigrams with zero counts according to the Good-Turing estimate.

The first constraint corresponds to the following equation:

$$d_r = \mu \frac{r^*}{r} \tag{11.29}$$

for $r \in \{1,\ldots k\}$ with some constant μ. The Good-Turing estimate predicts that the total mass assigned to bigrams with zero counts is $n_0 \frac{n_1}{n_0} = n_1$, and the second constraint corresponds to the equation

$$\sum_{r=1}^{k} n_r (1-d_r) r = n_1 \tag{11.30}$$

Based on Eq. (11.30), the unique solution is given by:

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}} \tag{11.31}$$

Katz smoothing for higher-order n-gram models is defined analogously. The Katz n-gram backoff model is defined in terms of the Katz (n-1)-gram model. To end the recursion, the Katz unigram model is taken to be the maximum likelihood unigram model. It is usually necessary to smooth n_r when using the Good-Turing estimate, e.g., for those n_r that are very low. However, in Katz smoothing this is not essential because the Good-Turing estimate is used only for small counts $r<=k$, and n_r is generally fairly high for these values of r. The procedure of Katz smoothing can be summarized as in Algorithm 11.2.

In fact, the Katz backoff model can be expressed in terms of the interpolated model defined in Eq. (11.23), in which the interpolation weight is obtained via Eq. (11.26) and (11.27).

ALGORITHM 11.2: *KATZ SMOOTHING*

$$P_{Katz}(w_i \mid w_{i-1}) = \begin{cases} C(w_{i-1}w_i)/C(w_{i-1}) & \text{if } r > k \\ d_r C(w_{i-1}w_i)/C(w_{i-1}) & \text{if } k \geq r > 0 \\ \alpha(w_{i-1})P(w_i) & \text{if } r = 0 \end{cases}$$

where $d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}$ and $\alpha(w_{i-1}) = \frac{1 - \sum_{w_i : r>0} P_{Katz}(w_i \mid w_{i-1})}{1 - \sum_{w_i : r>0} P(w_i)}$

11.4.2.2. Alternative Backoff Models

In a similar manner to the Katz backoff model, there are other ways to discount the probability mass. For instance, *absolute discounting* involves subtracting a fixed discount $D <= 1$ from each nonzero count. If we express the absolute discounting in term of interpolated models, we have the following:

$$P_{abs}(w_i|w_{i-n+1}\cdots w_{i-1}) = \frac{\max\{C(w_{i-n+1}\cdots w_i)-D,0\}}{\sum_{w_i} C(w_{i-n+1}\cdots w_i)} + (1-\lambda_{w_{i-n+1}\cdots w_{i-1}})P_{abs}(w_i|w_{i-n+2}\cdots w_{i-1}) \quad (11.32)$$

To make this distribution sum to 1, we normalize it to determine $\lambda_{w_{i-n+1}\cdots w_{i-1}}$. Absolute discounting is explained with the Good-Turing estimate. Empirically the average Good-Turing discount $r-r^*$ associated with n-grams of larger counts (r over 3) is largely constant over r.

Consider building a bigram model on data where there exists a word that is very common, say *Francisco*, that occurs only after a single word, say *San*. Since *C(Francisco)* is high, the unigram probability *P(Francisco)* will be high, and an algorithm such as absolute discounting or Katz smoothing assigns a relatively high probability to occurrence of the word *Francisco* after novel bigram histories. However, intuitively this probability should not be high, since in the training data the word *Francisco* follows only a single history. That is, perhaps *Francisco* should receive a low unigram probability, because the only time the word occurs is when the last word is *San*, in which case the bigram probability models its probability well.

Extending this line of reasoning, perhaps the unigram probability used should not be proportional to the number of occurrences of a word, but instead to the number of different words that it follows. To give an intuitive argument, imagine traversing the training data sequentially and building a bigram model on the preceding data to predict the current word. Then, whenever the current bigram does not occur in the preceding data, the unigram probability becomes a large factor in the current bigram probability. If we assign a count to the corresponding unigram whenever such an event occurs, then the number of counts assigned to each unigram is simply the number of different words that it follows. In Kneser-Ney smoothing [40], the lower-order n-gram is not proportional to the number of occurrences of a word, but instead to the number of *different* words that it follows. We summarize the Kneser-Ney backoff model in Algorithm 11.3.

Kneser-Ney smoothing is an extension of other backoff models. Most of the previous models used the lower-order n-grams trained with ML estimation. Kneser-Ney smoothing instead considers a lower-order distribution as a significant factor in the combined model such that they are optimized together with other parameters. To derive the formula, more generally, we express it in terms of the interpolated model specified in Eq. (11.23) as:

$$P_{KN}(w_i \mid w_{i-n+1}...w_{i-1})$$
$$= \frac{\max\{C(w_{i-n+1}...w_i) - D, 0\}}{\sum_{w_i} C(w_{i-n+1}...w_i)} + (1 - \lambda_{w_{i-n+1}...w_{i-1}}) P_{KN}(w_i \mid w_{i-n+2}...w_{i-1}) \quad (11.33)$$

To make this distribution sum to 1, we have:

$$1 - \lambda_{w_{i-n+1}...w_{i-1}} = \frac{D}{\sum_{w_i} C(w_{i-n+1}...w_i)} \mathbb{C}(w_{i-n+1}...w_{i-1}\bullet) \quad (11.34)$$

where $\mathbb{C}(w_{i-n+1}...w_{i-1}\bullet)$ is the number of unique words that follow the history $w_{i-n+1}...w_{i-1}$. This equation enables us to interpolate the lower-order distribution with all words, not just with words that have zero counts in the higher-order distribution.

ALGORITHM 11.3: *KNESER-NEY BIGRAM SMOOTHING*

$$P_{KN}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{\max\{C(w_{i-1}w_i) - D, 0\}}{C(w_{i-1})} & \text{if } C(w_{i-1}w_i) > 0 \\ \alpha(w_{i-1}) P_{KN}(w_i) & \text{otherwise} \end{cases}$$

where $P_{KN}(w_i) = \mathbb{C}(\bullet w_i) / \sum_{w_i} \mathbb{C}(\bullet w_i)$, $\mathbb{C}(\bullet w_i)$ is the number of unique words preceding w_i.

$\alpha(w_{i-1})$ is chosen to make the distribution sum to 1 so that we have:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i : C(w_{i-1}w_i) > 0} \dfrac{\max\{C(w_{i-1}w_i) - D, 0\}}{C(w_{i-1})}}{1 - \sum_{w_i : C(w_{i-1}w_i) > 0} P_{KN}(w_i)}$$

Now, take the bigram case as an example. We need to find a unigram distribution $P_{KN}(w_i)$ such that the marginal of the bigram smoothed distributions should match the marginal of the training data:

$$\frac{C(w_i)}{\sum_{w_i} C(w_i)} = \sum_{w_{i-1}} P_{KN}(w_{i-1}w_i) = \sum_{w_{i-1}} P_{KN}(w_i \mid w_{i-1}) P(w_{i-1}) \quad (11.35)$$

For $P(w_{i-1})$, we simply take the distribution found in the training data

$$P(w_{i-1}) = \frac{C(w_{i-1})}{\sum_{w_{i-1}} C(w_{i-1})} \quad (11.36)$$

We substitute Eq. (11.33) in Eq. (11.35). For the bigram case, we have:

$$
\begin{aligned}
& C(w_i) \\
& = \sum_{w_{i-1}} C(w_{i-1})[\frac{\max\{C(w_{i-1}w_i)-D,0\}}{\sum_{w_i} C(w_{i-1}w_i)} + \frac{D}{\sum_{w_i} C(w_{i-1}w_i)} \mathbb{C}(w_{i-1}\bullet) P_{KN}(w_i)] \\
& = \sum_{w_{i-1}:C(w_{i-1}w_i)>0} C(w_{i-1}) \frac{C(w_{i-1}w_i)-D}{C(w_{i-1})} + \sum_{w_{i-1}} C(w_{i-1}) \frac{D}{C(w_{i-1})} \mathbb{C}(w_{i-1}\bullet) P_{KN}(w_i) \\
& = C(w_i) - \mathbb{C}(\bullet w_{i-1})D + DP_{KN}(w_i) + DP_{KN}(w_i) \sum_{w_{i-1}} \mathbb{C}(w_{i-1}\bullet)
\end{aligned}
\tag{11.37}
$$

Solving the equation, we get

$$
P_{KN}(w_i) = \frac{\mathbb{C}(\bullet w_i)}{\sum_{w_i} \mathbb{C}(\bullet w_i)} \tag{11.38}
$$

which can be generalized to higher-order models:

$$
P_{KN}(w_i \mid w_{i-n+2}...w_{i-1}) = \frac{\mathbb{C}(\bullet w_{i-n+2}...w_i)}{\sum_{w_i} \mathbb{C}(\bullet w_{i-n+2}...w_i)} \tag{11.39}
$$

where $\mathbb{C}(\bullet w_{i-n+2}...w_i)$ is the number of different words that precede $w_{i-n+2}...w_i$.

In practice, instead of using a single discount D for all nonzero counts as in Kneser-Ney smoothing, we can have a number of different parameters (D_i) that depend on the range of counts:

$$
\begin{aligned}
& P_{KN}(w_i \mid w_{i-n+1}...w_{i-1}) \\
& = \frac{C(w_{i-n+1}...w_i) - D(C(w_{i-n+1}...w_i))}{\sum_{w_i} C(w_{i-n+1}...w_i)} + \\
& + \gamma(w_{i-n+1}...w_{i-1}) P_{KN}(w_i \mid w_{i-n+2}...w_{i-1})
\end{aligned}
\tag{11.40}
$$

This modification is motivated by evidence that the ideal average discount for n-grams with one or two counts is substantially different from the ideal average discount for n-grams with higher counts.

11.4.3. Class *N*-grams

As discussed in Chapter 2, we can define classes for words that exhibit similar semantic or grammatical behavior. This is another effective way to handle the data sparsity problem.

Class-based language models have been shown to be effective for rapid adaptation, training on small data sets, and reduced memory requirements for real-time speech applications.

For any given assignment of a word w_i to class c_i, there may be many-to-many mappings, e.g., a word w_i may belong to more than one class, and a class c_i may contain more than one word. For the sake of simplicity, assume that a word w_i can be uniquely mapped to only one class c_i. The n-gram model can be computed based on the previous n-1 classes:

$$P(w_i|c_{i-n+1}\cdots c_{i-1}) = P(w_i \mid c_i)P(c_i|c_{i-n+1}\cdots c_{i-1}) \tag{11.41}$$

where $P(w_i|c_i)$ denotes the probability of word w_i given class c_i in the current position, and $P(c_i|c_{i-n+1}\cdots c_{i-1})$ denotes the probability of class c_i given the class history. With such a model, we can learn the class mapping $w \rightarrow c$ from either a training text or task knowledge we have about the application. In general, we can express the class trigram as:

$$P(\mathbf{W}) = \sum_{c_1 \cdots c_n} \prod_i P(w_i \mid c_i)P(c_i \mid c_{i-2}, c_{i-1}) \tag{11.42}$$

If the classes are nonoverlapping, i.e. a word may belong to only one class, then Eq. (11.42) can be simplified as:

$$P(\mathbf{W}) = \prod_i P(w_i \mid c_i)P(c_i \mid c_{i-2}, c_{i-1}) \tag{11.43}$$

If we have the mapping function defined, we can easily compute the class n-gram. We can estimate the empirical frequency of each word $C(w_i)$, and of each class $C(c_i)$. We can also compute the empirical frequency that a word from one class will be followed immediately by a word from another $C(c_{i-1}c_i)$. As a typical example, the bigram probability of a word given the prior word (class) can be estimated as

$$P(w_i \mid w_{i-1}) = P(w_i \mid c_{i-1}) = P(w_i \mid c_i)P(c_i \mid c_{i-1}) = \frac{C(w_i)}{C(c_i)} \frac{C(c_{i-1}c_i)}{C(c_{i-1})} \tag{11.44}$$

For general-purpose large vocabulary dictation applications, class-based n-grams have not significantly improved recognition accuracy. They are mainly used as a backoff model to complement the lower-order n-grams for better smoothing. Nevertheless, for limited domain speech recognition, the class-based n-gram is very helpful as the class can efficiently encode semantic information for improved key word spotting and speech understanding accuracy.

11.4.3.1. Rule-Based Classes

There are a number of ways to cluster words together based on the syntactic-semantic information that exists for the language and the task. For example, part-of-speech can be gen-

erally used to produce a small number of classes although this may lead to significantly increased perplexity. Alternatively, if we have domain knowledge, it is often advantageous to cluster together words that have a similar semantic functional role. For example, if we need to build a conversational system for air travel information systems, we can group the name of different airlines such as *United Airlines, KLM,* and *Air China*, into a broad *airline class.* We can do the same thing for the names of different airports such as *JFK, Narita,* and *Heathrow*, the names of different cities like *Beijing, Pittsburgh,* and *Moscow,* and so on. Such an approach is particularly powerful, since the amount of training data is always limited. With generalized broad classes of semantically interpretable meaning, it is easy to add a new airline such as *Redmond Air* into the classes if there is indeed a start-up airline named *Redmond Air* that the system has to incorporate. The system is now able to assign a reasonable probability to a sentence like "*Show me all flights of Redmond Air from Seattle to Boston*" in a similar manner as "*Show me all flights of United Airlines from Seattle to Boston.*" We only need to estimate the probability of *Redmond Air,* given the airline class c_i. We can use the existing class n-gram model that contains the broad structure of the air travel information system as it is.

Without such a broad interpretable class, it would be extremely difficult to deal with new names the system needs to handle, although these new names can always be mapped to the special class of the unknown word or proper noun classes. For these new words, we can alternatively map them into a word that has a similar syntactic and semantic role. Thus, the new word inherits all the possible word trigram relationships that may be very similar to those of the existing word observed with the training data.

11.4.3.2. Data-driven Classes

For a general-purpose dictation application, it is impractical to derive functional classes in the same manner as a domain-specific conversational system that focuses on a narrow task. Instead, data-driven clustering algorithms have been used to generalize the concept of word similarities, which is in fact a search procedure to find a class label for each word with a predefined objective function. The set of words with the same class label is called a cluster. We can use the maximum likelihood criterion as the objective function for a given training corpus and a given number of classes, which is equivalent to minimizing the perplexity for the training corpus. Once again, the EM algorithm can be used here. Each word can be initialized to a random cluster (class label). At each iteration, every word is moved to the class that produces the model with minimum perplexity [9, 48]. The perplexity modifications can be calculated independently, so that each word is evaluated as if all other word classes were held fixed. The algorithm converges when no single word can be moved to another class in a way that reduces the perplexity of the clustered n-gram model.

One special kind of class n-gram models is based on the decision tree as discussed in Chapter 4. We can use it to create equivalent classes for words in the history, so that can we

have a compact long-distance n-gram language model [2]. The sequential decomposition, as expressed in Eq. (11.12), is approximated as:

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i | E(w_1, w_2, \ldots, w_{i-1})) = \prod_{i=1}^{n} P(w_i | E(\mathbf{h})) \tag{11.45}$$

where $E(\mathbf{h})$ denotes a many-to-one mapping function that groups word histories $\mathbf{h}$ into some equivalence classes. It is important to have a scheme that can provide adequate information about the history so it can serve as a basis for prediction. In addition, it must yield a set of classes that can be reliably estimated. The decision tree method uses entropy as a criterion in developing the equivalence classes that can effectively incorporate long-distance information. By asking a number of questions associated with each node, the decision tree can classify the history into a small number of equivalence classes. Each leaf of the tree, thus, has the probability $P(w_i | E(w_1 \ldots w_{i-1}))$ that is derived according to the number of times the word w_i is found in the leaf. The selection of questions in building the tree can be infinite. We can consider not only the syntactic structure, but also semantic meaning to derive permissible questions from which the entropy criterion would choose. A full-fledged question set that is based on detailed analysis of the history is beyond the limit of our current computing resources. As such, we often use the membership question to check each word in the history.

11.4.4. Performance of *N*-gram Smoothing

The performance of various smoothing algorithms depends on factors such as the training-set sizes. There is a strong correlation between the test-set perplexity and word error rate. Smoothing algorithms leading to lower perplexity generally result in a lower word error rate. Among all the methods discussed here, the Kneser-Ney method slightly outperforms other algorithms over a wide range of training-set sizes and corpora, and for both bigram and trigram models. Albeit the difference is not large, the good performance of the Kneser-Ney smoothing is due to the modified backoff distributions. The Katz algorithms and deleted interpolation smoothing generally yield the next best performance. All these three smoothing algorithms perform significantly better than the n-gram model without any smoothing. The deleted interpolation algorithm performs slightly better than the Katz method in sparse data situations, and the reverse is true when data are plentiful. Katz's algorithm is particularly good at smoothing larger counts; these counts are more prevalent in larger data sets.

Class n-grams offer different kind of smoothing. While clustered n-gram models often offer no significant test-set perplexity reduction in comparison to the word n-gram model, it is beneficial to smooth the word n-gram model via either backoff or interpolation methods.

For example, the decision-tree based long-distance class language model does not offer significantly improved speech recognition accuracy until it is interpolated with the word trigram. They are effective as a domain-specific language model if the class can accommodate domain-specific information.

Smoothing is a fundamental technique for statistical modeling, important not only for language modeling but for many other applications as well. Whenever data sparsity is an issue, smoothing can help performance, and data sparsity is almost always an issue in statistical modeling. In the extreme case, where there is so much training data that all parameters can be accurately trained without smoothing, you can almost always expand the model, such as by moving to a higher-order n-gram model, to achieve improved performance. With more parameters, data sparsity becomes an issue again, but a proper smoothing model is usually more accurate than the original model. Thus, no matter how much data you have, smoothing can almost always help performance, and for a relatively small effort.

11.5. ADAPTIVE LANGUAGE MODELS

Dynamic adjustment of the language model parameter, such as n-gram probabilities, vocabulary size, and the choice of words in the vocabulary, is important, since the topic of conversation is highly nonstationary [4, 33, 37, 41, 46]. For example, in a typical dictation application, a particular set of words in the vocabulary may suddenly burst forth and then become dormant later, based on the current conversation. Because the topic of the conversation may change from time to time, the language model should be dramatically different based on the topic of the conversation. We discuss several adaptive techniques that can improve the quality of the language model based on the real usage of the application.

11.5.1. Cache Language Models

To adjust word frequencies observed in the current conversation, we can use a dynamic *cache* language model [41]. The basic idea is to accumulate word n-grams dictated so far in the current document and use these to create a local dynamic n-gram model such as bigram $P_{cache}(w_i|w_{i-1})$. Because of limited data and nonstationary nature, we should use a lower-order language model that is no higher than a trigram model $P_{cache}(w_i|w_{i-2}w_{i-1})$, which can be interpolated with the dynamic bigram and unigram. Empirically, we need to normally give a high weight to the unigram cache model, because it is better trained with the limited data in the cache.

With the cache trigram, we interpolate it with the static n-gram model $P_s(w_i|w_{i-n+1}\cdots w_{i-1})$. The interpolation weight can be made to vary with the size of the cache.

$$\begin{aligned} &P_{cache}(w_i \mid w_{i-n+1}\cdots w_{i-1}) \\ &= \lambda_c P_s(w_i \mid w_{i-n+1}\cdots w_{i-1}) + (1-\lambda_c) P_{cache}(w_i \mid w_{i-2}w_{i-1}) \end{aligned} \tag{11.46}$$

The cache model is desirable in practice because of its impressive empirical performance improvement. In a dictation application, we often encounter new words that are not in the static vocabulary. The same words also tend to be repeated in the same article. The cache model can address this problem effectively by adjusting the parameters continually as recognition and correction proceed for incrementally improved performance. A noticeable benefit is that we can better predict words belonging to fixed phrases such as *Windows NT* and *Bill Gates*.

11.5.2. Topic-Adaptive Models

The topic can change over time. Such topic or style information plays a critical role in improving the quality of the static language model. For example, the prediction of whether the word following the phrase *the operating* is *system* or *table* can be improved substantially by knowing whether the topic of discussion is related to computing or medicine.

Domain or topic-clustered language models split the language model training data according to topic. The training data may be divided using the known category information or using automatic clustering. In addition, a given segment of the data may be assigned to multiple topics. A topic-dependent language model is then built from each cluster of the training data. Topic language models are combined using linear interpolation or other methods such as maximum entropy techniques discussed in Section 11.5.3.

We can avoid any pre-defined clustering or segmentation of the training data. The reason is that the best clustering may become apparent only when the current topic of discussion is revealed. For example, when the topic is hand-injury to baseball player, the pre-segmented clusters of topic *baseball & hand-injuries* may have to be combined. This leads to a union of the two clusters, whereas the ideal dataset is obtained by the intersection of these clusters. In general, various combinations of topics lead to a combinatorial explosion in the number of compound topics, and it appears to be a difficult task to anticipate all the needed combinations beforehand.

We base our determination of the most suitable language model data to build a model upon the particular history of a given document. For example, we can use it as a query against the entire training database of documents using *information retrieval* techniques [57]. The documents in the database can be ranked by relevance to the query. The most relevant documents are then selected as the adaptation set for the topic-dependent language model. The process can be repeated as the document is updated.

There are two major steps we need to consider here. The first involves using the available document history to retrieve similar documents from the database. The second consists of using the similar document set retrieved in the first step to adapt the general or topic-independent language model. Available document history depends upon the design and the requirements of the recognition system. If the recognition system is designed for live-mode application, where the recognition results must be presented to the user with a small delay, the available document history will be the history of the document user created so far. On the other hand, in a recognition system designed for batch operation, the amount of time

taken by the system to recognize speech is of little consequence to the user. In the batch mode, therefore, a multi-pass recognition system can be used, and the document history will be the recognizer transcript produced in the current pass.

The well-known information retrieval measure called *TFIDF* can be used to locate similar documents in the training database [57]. The term frequency (TF) tf_{ij} is defined as the frequency of the jth term in the document D_i, the unigram count of the term j in the document D_i. The inverse document frequency (IDF) idf_j is defined as the frequency of the jth term over the entire database of documents, which can be computed as:

$$idf_j = \frac{\text{Total number of documents}}{\text{Number of documents containing term } j} \tag{11.47}$$

The combined TF-IDF measure is defined as:

$$TFIDF_{ij} = tf_{ij} \log(idf_i) \tag{11.48}$$

The combination of TF and IDF can help to retrieve similar documents. It highlights words of particular interest to the query (via TF), while de-emphasizing common words that appear across different documents (via IDF). Each document including the query itself, can be represented by the TFIDF vector. Each element of the vector is the TFIDF value that corresponds to a word (or a term) in the vocabulary. Similarity between the two documents is then defined to be the cosine of the angle between the corresponding vectors. Therefore, we have:

$$Similarity(D_i, D_j) = \frac{\sum_k tfidf_{ik} * tfidf_{jk}}{\sqrt{\sum_k (tfidf_{ik})^2 * \sum_k (tfidf_{jk})^2}} \tag{11.49}$$

All the documents in the training database are ranked by the decreasing similarity between the document and the history of the current document dictated so far, or by a topic of particular interest to the user. The most similar documents are selected as the adaptation set for the topic-adaptive language model [46].

11.5.3. Maximum Entropy Models

The language model we have discussed so far combines different n-gram models via linear interpolation. A different way to combine sources is the maximum entropy approach. It constructs a single model that attempts to capture all the information provided by the various knowledge sources. Each such knowledge source is reformulated as a set of constraints that the desired distribution should satisfy. These constraints can be, for example, marginal distributions of the combined model. Their intersection, if not empty, should contain a set of

probability functions that are consistent with these separate knowledge sources. Once the desired knowledge sources have been incorporated, we make no other assumption about other constraints, which leads to choosing the flattest of the remaining possibilities, the one with the highest entropy. The maximum entropy principle can be stated as follows:

- Reformulate different information sources as constraints to be satisfied by the target estimate.
- Among all probability distributions that satisfy these constraints, choose the one that has the highest entropy.

Given a general event space $\{\mathbf{X}\}$, let $P(\mathbf{X})$ denote the combined probability function. Each constraint is associated with a characteristic function of a subset of the sample space, $f_i(\mathbf{X})$. The constraint can be written as:

$$\sum_{\mathbf{X}} P(\mathbf{X}) f_i(\mathbf{X}) = E_i \tag{11.50}$$

where E_i is the corresponding desired expectation for $f_i(\mathbf{X})$, typically representing the required marginal probability of $P(\mathbf{X})$. For example, to derive a word trigram model, we can reformulate Eq. (11.50) so that constraints are introduced for unigram, bigram, and trigram probabilities. These constraints are usually set only where marginal probabilities can be estimated from a corpus. For example, the unigram constraint can be expressed as

$$f_{w_1}(w) = \begin{cases} 1 & \text{if } w=w_1 \\ 0 & \text{otherwise} \end{cases} \tag{11.51}$$

The desired value E_{w_1} can be the empirical expectation in the training data, $\sum_{w \in \text{training data}} f_{w_1}(w)/N$, and the associated constraint is

$$\sum_{\mathbf{h}} P(\mathbf{h}) \sum_{w} P(w \mid \mathbf{h}) f_{w_1}(w) = E_{w_1} \tag{11.52}$$

where $\mathbf{h}$ is the word history preceding word w.

We can choose $P(\mathbf{X})$ to diverge minimally from some other known probability function $Q(\mathbf{X})$, that is, to minimize the divergence function:

$$\sum_{\mathbf{X}} P(\mathbf{X}) \log \frac{P(\mathbf{X})}{Q(\mathbf{X})} \tag{11.53}$$

When $Q(\mathbf{X})$ is chosen as the uniform distribution, the divergence is equal to the negative of entropy with a constant. Thus minimizing the divergence function leads to maximiz-

ing the entropy. Under a minor consistent assumption, a unique solution is guaranteed to exist in the form [20]:

$$P(\mathbf{X}) \propto \prod_i \mu_i^{f_i(\mathbf{X})} \tag{11.54}$$

where μ_i is an unknown constant to be found. To search the exponential family defined by Eq. (11.54) for the μ_i that make $P(\mathbf{X})$ satisfy all the constraints, an iterative algorithm called generalized iterative scaling exists [20]. It guarantees to converge to the solution with some arbitrary initial μ_i. Each iteration creates a new estimate $P(\mathbf{X})$, which is improved in the sense that it matches the constraints better than its previous iteration [20]. One of the most effective applications of the maximum entropy model is to integrate the cache constraint into the language model directly, instead of interpolating the cache n-gram with the static n-gram. The new constraint is that the marginal distribution of the adapted model is the same as the lower-order n-gram in the cache [56]. In practice, the maximum entropy method has not offered any significant improvement in comparison to the linear interpolation.

11.6. PRACTICAL ISSUES

In a speech recognition system, every string of words $\mathbf{W} = w_1 w_2 \ldots w_n$ taken from the prescribed vocabulary can be assigned a probability, which is interpreted as the a priori probability to guide the recognition process and is a contributing factor in the determination of the final transcription from a set of partial hypothesis. Without language modeling, the entire vocabulary must be considered at every decision point. It is impossible to eliminate many candidates from consideration, or alternatively to assign higher probabilities to some candidates than others to considerably reduce recognition costs and errors.

11.6.1. Vocabulary Selection

For most speech recognition systems, an inflected form is considered as a different word. This is because these inflected forms typically have different pronunciations, syntactic roles, and usage patterns. So the words *work*, *works*, *worked*, and *working* are counted as four different words in the vocabulary.

We prefer to have a smaller vocabulary size, since this eliminates potential confusable candidates in speech recognition, leading to improved recognition accuracy. However, the limited vocabulary size imposes a severe constraint on the users and makes the system less flexible. In practice, the percentage of the Out-Of-Vocabulary (OOV) word rate directly affects the perceived quality of the system. Thus, we need to balance two kinds of errors, the OOV rate and the word recognition error rate. We can have a larger vocabulary to minimize the OOV rate if the system resources permit. We can minimize the expected OOV rate of the

test data with a given vocabulary size. A corpus of text is used in conjunction with dictionaries to determine appropriate vocabularies.

The availability of various types and amounts of training data, from various time periods, affects the quality of the derived vocabulary. Given a collection of training data, we can create an ordered word list with the lowest possible OOV curve, such that, for any desired vocabulary size V, a minimum-OOV-rate vocabulary can be derived by taking the most frequent V words in that list. Viewed this way, the problem becomes one of estimating unigram probabilities of the test distribution, and then ordering the words by these estimates.

As illustrated in Figure 11.6, the perplexity generally increases with the vocabulary size, albeit it really does not make much sense to compare the perplexity of different vocabulary sizes. There are generally more competing words for a given context when the vocabulary size becomes big, which leads to increased recognition error rate. In practice, this is offset by the OOV rate, which decreases with the vocabulary size as illustrated in Figure 11.7. If we keep the vocabulary size fixed, we need more than 200,000 words in the vocabulary to have 99.5% English words coverage. For more inflectional languages such as German, larger vocabulary sizes are required to achieve coverage similar to that of English.[6]

In practice, it is far more important to use data from a specific topic or domain, if we know in what domain the speech recognizer is used. In general, it is also important to consider coverage of a specific time period. We should use training data from that period, or as close to it as possible. For example, if we know we will talk only about air travel, we benefit from using the air-travel related vocabulary and language model. This point is well illustrated by the fact that the perplexity of the domain-dependent bigram can be reduced by more than a factor of five over the general-purpose English trigram.

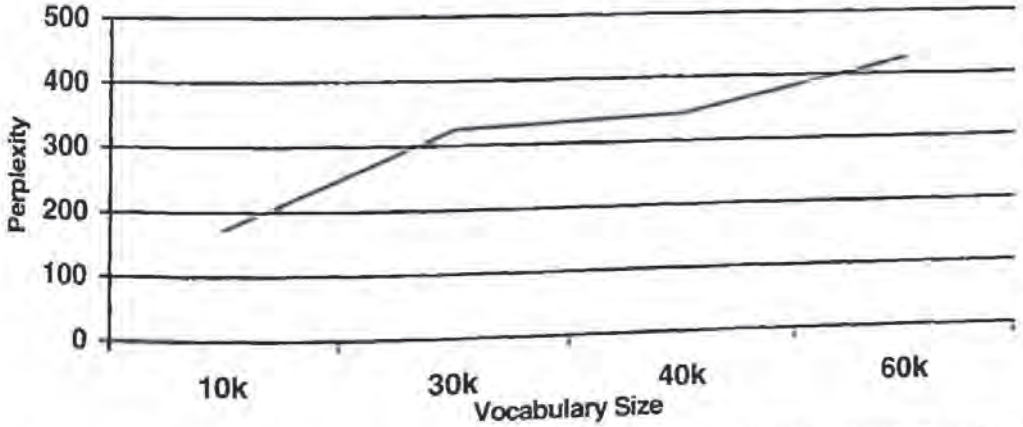


Figure 11.6 The perplexity of bigram with different vocabulary sizes. The training set consists of 500 million words derived from various sources, including newspapers and email. The test set comes from the whole Microsoft Encarta, an encyclopedia that has a wide coverage of different topics.

[6] The OOV rate of German is about twice as high as that of English with a 20k-word vocabulary [34].

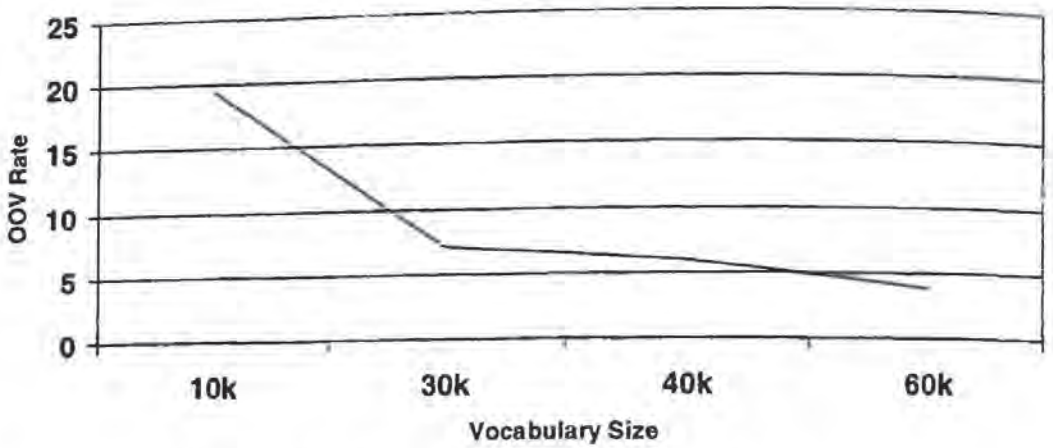


Figure 11.7 The OOV rate with different vocabulary size. The training set consists of 500 million words derived from various sources including newspaper and email. The test set came from the whole Microsoft Encarta encyclopedia.

For a user of a speech recognition system, a more personalized vocabulary can be much more effective than a general fixed vocabulary. The coverage can be dramatically improved as customized new words are added to a starting static vocabulary of 20,000. Typically, the coverage of such a system can be improved from 93% to more than 98% after 1000-4000 customized words are added to the vocabulary [18].

In North American general business English, the least frequent words among the most frequent 60,000 have a frequency of about 1:7,000,000. In optimizing a 60,000-word vocabulary we need to distinguish words with frequency of 1:7,000,000 from those that are slightly less frequent. To differentiate somewhat reliably between a 1:7,000,000 word and, say, a 1:8,000,000 word, we need to observe them enough times for the difference in their counts to be statistically reliable. For constructing a decent vocabulary, it is important that most such words are ranked correctly. We may need 100,000,000 words to estimate these parameters. This agrees with the empirical results, in which as more training data is used, the OOV curve improves rapidly up to 50,000,000 words and then more slowly beyond that point.

11.6.2. *N*-gram Pruning

When high order n-gram models are used, the model sizes typically become too large for practical applications. It is necessary to prune parameters from n-gram models such that the relative entropy between the original and the pruned model is minimized. You can choose n-grams so as to maximize performance (i.e., minimize perplexity) while minimizing the model size [39, 59, 64].

The criterion to prune n-grams can be based on some well-understood information-theoretic measure of language model quality. For example, the pruning method by Stolcke [64] removes some n-gram estimates while minimizing the performance loss. After pruning,

the retained explicit n-gram probabilities are unchanged, but backoff weights are recomputed. Stolcke pruning uses the criterion that minimizes the distance between the distribution embodied by the original model and that of the pruned model based on the *Kullback-Leibler distance* defined in Eq. (3.181). Since it is infeasible to maximize over all possible subsets of n-grams, Stolcke prunning assumes that the n-grams affect the relative entropy roughly independently, and compute the distance due to each individual n-gram. The n-grams are thus ranked by their effect on the model entropy, and those that increase relative entropy the least are pruned accordingly. The main approximation is that we do not consider possible interactions between selected n-grams, and prune based solely on relative entropy due to removing a single n-gram. This avoids searching the exponential space of n-gram subsets.

To compute the relative entropy, $KL(p \parallel p')$, between the original and pruned n-gram models p and p', there is no need to sum over the vocabulary. By plugging in the terms for the backoff estimates, the sum can be factored as shown in Eq. (11.55) for a more efficient computation.

$$\begin{aligned} KL(p \parallel p') = & -P(h)\{P(w \mid h)[\log P(w \mid h') + \log \alpha'(h) - \log P(w \mid h)] \\ & + [\log \alpha'(h) - \log \alpha(h)](1 - \sum_{w_i \in \neg Backoff(w,h)} P(w_i \mid h))\} \end{aligned} \tag{11.55}$$

where the sum in $\sum_{w_i \in \neg Backoff(w,h)} P(w_i \mid h)$ is over all non-backoff estimates. To compute the revised backoff weights $\alpha'(h)$, you can simply drop the term for the pruned n-gram from the summation (backoff weight computation is illustrated in Algorithm 11.1).

In practice, pruning is highly effective. Stolcke reported that the trigram model can be compressed by more than 25% without degrading recognition performance. Comparing the pruned 4-gram model to the unpruned trigram model, it is better to use pruned 4-grams than to use a much larger number of trigrams.

11.6.3. CFG vs. N-gram Models

This chapter has discussed two major language models. While CFGs remain one of the most important formalisms for interpreting natural language, word n-gram models are surprisingly powerful for domain-independent applications. These two formalisms can be unified for both speech recognition and spoken language understanding. To improve portability of the domain-independent n-gram, it is possible to incorporate domain-specific CFGs into the domain-independent n-gram that can improve generalizability of the CFG and specificity of the n-gram.

The CFG is not only powerful enough to describe most of the structure in spoken language, but also restrictive enough to have efficient parsers. $P(\mathbf{W})$ is regarded as 1 or 0 depending upon whether the word sequence is accepted or rejected by the grammar. The

problem is that the grammar is *almost always incomplete*. A CFG-based system is good only when you know what sentences to speak, which diminishes the system's value and usability of the system. The advantage of CFG's structured analysis is, thus, nullified by the poor coverage in most real applications. On the other hand, the *n*-gram model is trained with a large amount of data, the *n*-word dependency can often accommodate both syntactic and semantic structure seamlessly. The prerequisite of this approach is that we have enough training data. The problem for *n*-gram models is that we need a lot of data and the model may not be specific enough.

It is possible to take advantage of both rule-based CFGs and data-driven *n*-grams. Let's consider the following training sentences:

```
Meeting at three with Zhou Li.
Meeting at four PM with Derek.
```

If we use a word trigram, we estimate *P(Zhou|three with)* and *P(Derek|PM with)*, etc. There is no way we can capture needed long-span semantic information in the training data. A unified model has a set of CFGs that can capture the semantic structure of the domain. For the example listed here, we have a CFG for {name} and {time}, respectively. We can use the CFG to parse the training data to spot all the potential semantic structures in the training data. The training sentences now look like:

```
Meeting {at three:TIME} with {Zhou Li:NAME}
Meeting {at four PM:TIME} with {Derek: NAME}
```

With analyzed training data, we can estimate our *n*-gram probabilities as usual. We have probabilities, such as P({name}|{time} with), instead of *P(Zhou|three with)*, which is more meaningful and accurate. Inside each CFG we also derive P("*Zhou Li*"|{name}) and P("*four PM*"|{time}) from the existing *n*-gram (*n*-gram probability inheritance) so that they are normalized. If we add a new name to the existing {name} CFG, we use the existing *n*-gram probabilities to renormalize our CFGs for the new name. The new approach can be regarded as a standard *n*-gram in which the vocabulary consists of words and structured classes, as discussed in Section 11.4.3. The structured class can be very simple, such as {date}, {time}, and {name}, or can be very complicated, such as a CFG that contains deep structured information. The probability of a word or class depends on the previous words or CFG classes.

It is possible to inherit probability from a word *n*-gram LM. Let's take word trigram as our example here. An input utterance $\mathbf{W} = w_1 w_2 ... w_n$ can be segmented into a sequence $\mathbf{T} = t_1 t_2 ... t_m$, where each t_i is either a word in $\mathbf{W}$ or a CFG non-terminal that covers a sequence of words $\overline{u}_{t_i}$ in $\mathbf{W}$. The likelihood of $\mathbf{W}$ under the segmentation $\mathbf{T}$ is, therefore,

$$P(\mathbf{W},\mathbf{T}) = \prod_i P(t_i \mid t_{i-1}, t_{i-2}) \prod_i P(\overline{u}_{t_i} \mid t_i) \tag{11.56}$$

$P(\bar{u}_{t_i} | t_i)$, the likelihood of generating a word sequence $\bar{u}_{t_i} = [u_{t_i,1} u_{t_i,2} ... u_{t_i,k}]$ from the CFG non-terminal t_i, can be inherited from the domain-independent word trigram. We can essentially use the CFG constraint to condition the domain-independent trigram into a domain-specific trigram. Such a unified language model can dramatically improve cross-domain performance using domain-specific CFGs [66].

In summary, the CFG is widely used to specify the permissible word sequences in natural language processing when training corpora are unavailable. It is suitable for dealing with structured command and control applications in which the vocabulary is small and the semantics of the task is well defined. The CFG either accepts the input sentence or rejects it. There is a serious coverage problem associated with CFGs. In other words, the accuracy for the CFG can be extremely high when the test data are covered by the grammar. Unfortunately, unless the task is narrow and well-defined, most users speak sentences that may not be accepted by the CFG, leading to word recognition errors.

Statistical language models such as trigrams assign an estimated probability to any word that can follow a given word history without parsing the structure of the history. Such an approach contains some limited syntactic and semantic information, but these probabilities are typically trained from a large corpus. Speech recognition errors are much more likely to occur within trigrams and (especially) bigrams that have not been observed in the training data. In these cases, the language model typically relies on lower-order statistics. Thus, increased n-gram coverage translates directly into improved recognition accuracy, but usually at the cost of increased memory requirements.

It is interesting to compute the true entropy of the language so that we understand what a solid lower bound is for the language model. For English, Shannon [60] used human subjects to guess letters by looking at how many guesses it takes people to derive the correct one based on the history. We can thus estimate the probability of the letters and hence the entropy of the sequence. Shannon computed the per-letter entropy of English with an entropy of 1.3 bits for 26 letters plus space. This may be an underestimate, since it is based on a single text. Since the average length of English written words (including space) is about 5.5 letters, the Shannon estimate of 1.3 bits per letter corresponds to a per-word perplexity of 142 for general English.

Table 11.2 summarizes the performance of several different n-gram models on a 60,000-word continuous speech dictation application. The experiments used about 260 million words from a newspaper such as the *Wall Street Journal*. The speech recognizer is based on Whisper described in Chapter 9. As you can see from the table, when the amount of training data is sufficient, both Katz and Kneser-Ney smoothing offer comparable recognition performance, although Kneser-Ney smoothing offers a modest improvement when the amount of training data is limited.

In comparison to Shannon's estimate of general English word perplexity, the trigram language for the *Wall Street Journal* is lower (91.4 vs. 142). This is because the text is mostly business oriented with a fairly homogeneous style and word usage pattern. For example, if we use the trigram language for data from a new domain that is related to personal information management, the test-set word perplexity can increase to 378 [66].

Table 11.2 *N*-gram perplexity and its corresponding speaker-independent speech recognition word error rate.

Models	Perplexity	Word Error Rate
Unigram Katz	1196.45	14.85%
Unigram Kneser-Ney	1199.59	14.86%
Bigram Katz	176.31	11.38%
Bigram Kneser-Ney	176.11	11.34%
Trigram Katz	95.19	9.69%
Trigram Kneser-Ney	91.47	9.60%

11.7. HISTORICAL PERSPECTIVE AND FURTHER READING

There is a large and active area of research in both speech and linguistics. These two distinctive communities worked on the problem with very different paths, leading to the stochastic language models and the formal language theory. The linguistics community has developed tools for tasks like parsing sentences, assigning semantic relations to the parts of a sentence, and so on. Most of these parser algorithms have the same characteristics, that is, they tabulate each sub-derivation and reuse it in building any derivation that shares that sub-derivation with appropriate grammars [22, 65, 67]. They have polynomial complexity with respect to sentence length because of *dynamic programming* principles to search for optimal derivations with respect to appropriate evaluation functions on derivations. There are three well-known dynamic programming parsers with a worst-case behavior of $O(n^3)$, where n is the number of words in the sentence: the Cocke-Younger-Kasami (CYK) algorithm (a bottom-up parser, proposed by J. Cocke, D. Younger, and T. Kasami) [32, 67], the Graham-Harrison-Ruzzo algorithm (bottom-up) [30], and the Earley algorithm (top-down) [21].

On the other hand, the speech community has developed tools to predict the next word on the basis of what has been said, in order to improve speech recognition accuracy [35]. Neither approach has been completely successful. The formal grammar and the related parsing algorithms are too brittle for comfort and require a lot of human retooling to port from one domain to another. The lack of structure and deep understanding has taken its toll on statistical technology's ability to choose the right words to guide speech recognition.

In addition to those discussed in this chapter, many alternative formal techniques are available. Augmented context-free grammars are used for natural language to capture grammatical natural languages such as agreement and subcategorization. Examples include generalized phrase structure grammars and head-driven phrase structure grammars [26, 53]. You can further generalize the augmented context-free grammar to the extent that the requirement of *context free* becomes unnecessary. The entire grammar, known as the *unification grammar*, can be specified as a set of constraints between feature structures [62]. Most of these grammars have only limited success when applied to spoken language systems. In fact, no practical domain-independent parser of unrestricted text has been developed for spoken language systems, partly because disambiguation requires the specification of detailed semantic information. Analysis of the Susanne Corpus with a crude parser suggests

that over 80% of sentences are structurally ambiguous. More recently, large *treebanks* of parsed texts have given impetus to statistical approaches to parsing. Probabilities can be estimated from treebanks or plain text [6, 8, 24, 61] to efficiently rank analyses produced by modified chart parsing algorithms. These systems have yielded results of around 75% accuracy in assigning analyses to (unseen) test sentences from the same source as the unambiguous training material. Attempts have also been made to use statistical induction to *learn* the correct grammar for a given corpus of data [7, 43, 51, 58]. Nevertheless, these techniques are limited to simple grammars with category sets of a dozen or so non-terminals, or to training on manually parsed data. Furthermore, even when parameters of the grammar and control mechanism can be learned automatically from training corpora, the required corpora do not exist or are too small for proper training. In practice, we can devise grammars that specify directly how relationships relevant to the task may be expressed. For instance, one may use a phrase-structure grammar in which nonterminals stand for task concepts and relationships and rules specify possible expressions of those concepts and relationships. Such *semantic grammars* have been widely used for spoken language applications as discussed in Chapter 17.

It is worthwhile to point out that many natural language parsing algorithms are *NP-complete*, a term for a class of problems that are suspected to be particularly difficult to process. For example, maintaining lexical and agreement features over a potentially infinite-length sentence causes the unification-based formalisms to be NP-complete [3].

Since the predictive power of a general-purpose grammar is insufficient for reasonable performance, n-gram language models continue to be widely used. A complete proof of Good-Turing smoothing was presented by Church *et al.* [17]. Chen and Goodman [13] provide a detailed study on different n-gram smoothing algorithms. Jelinek's Eurospeech tutorial paper [35] provides an interesting historical perspective on the community's efforts to improve trigrams. Mosia and Giachin's paper [48] has detailed experimental results on class-based language models. Class-based model may be based on parts of speech or morphology [10, 16, 23, 47, 63]. More detailed discussion of the maximum entropy language model can be found in [5, 36, 42, 44, 52, 55, 56].

One interesting research area is to combine both n-grams and the structure that is present in language. A concerted research effort to explore structure-based language model may be the key for significant progress to occur in language modeling. This can be done as annotated data becomes available. Nasr *et al.* [50] have considered a new unified language model composed of several local models and a general model linking the local models together. The local model used in their system is based on the stochastic FSA, which is estimated from the training corpora. Other efforts to incorporate structured information are described in [12, 25, 27, 49, 66].

You can find tools to build n-gram language models at the CMU open source Web site[7] and SRI's language modeling toolkit Web site.[8] Both contain language modeling toolkits and documentation.

[7] http://www.speech.cs.cmu.edu/sphinx/
[8] http://www.speech.sri.com/projects/srilm/download.html

REFERENCES

[1] Aho, A.V. and J.D. Ullman, *The Theory of Parsing, Translation and Compiling*, 1972, Englewood Cliffs, NJ, Prentice-Hall.

[2] Bahl, L.R., *et al.*, "A Tree-Based Statistical Language Model for Natural Language Speech Recognition," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 1989, **37**(7), pp. 1001-1008.

[3] Barton, G., R. Berwick, and E. Ristad, *Computational Complexity and Natural Language*, 1987, Cambridge, MA, MIT Press.

[4] Bellegarda, J., "A Latent Semantic Analysis Framework for Large-Span Language Modeling," *Eurospeech*, 1997, Rhodes, Greece, pp. 1451-1454.

[5] Berger, A., S. DellaPietra, and V. DellaPietra, "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistics*, 1996, **22**(1), pp. 39-71.

[6] Black, E., *et al.*, "Towards History-based Grammars: Using Richer Models for Probabilistic Parsing," *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 1993, Columbus, Ohio, USA, pp. 31-37.

[7] Briscoe, E.J., ed. *Prospects for Practical Parsing: Robust Statistical Techniques*, in Corpus-based Research into Language: A Feschrift for Jan Aarts, ed. P.d. Haan and N. Oostdijk, 1994, Amsterdam. 67-95, Rodopi.

[8] Briscoe, E.J. and J. Carroll, "Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-based Grammars," *Computational Linguistics*, 1993, **19**, pp. 25-59.

[9] Brown, P.F., *et al.*, "Class-Based *N*-gram Models of Natural Language," *Computational Linguistics*, 1992(4), pp. 467-479.

[10] Cerf-Danon, H. and M. El-Bèze, "Three Different Probabilistic Language Models: Comparison and Combination," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, Canada, pp. 297-300.

[11] Charniak, E., "Statistical Parsing with a Context-Free Grammar and Word Statistics," *AAAI-97*, 1997, Menlo Park, pp. 598-603.

[12] Chelba, C., A. Corazza, and F. Jelinek, "A Context Free Headword Language Model" in *Proc. of IEEE Automatic Speech Recognition Workshop"* 1995, Snowbird, Utah, pp. 89-90.

[13] Chen, S. and J. Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling," *Proc. of Annual Meeting of the ACL*, 1996, Santa Cruz, CA.

[14] Chomsky, N., *Syntactic Structures*, 1957, The Hague: Mouton.

[15] Chomsky, N., *Aspects of the Theory of Syntax*, 1965, Cambridge, MIT Press.

[16] Church, K., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," *Proc. of 2nd Conf. on Applied Natural Language Processing*, 1988, Austin, Texas, pp. 136-143.

[17] Church, K.W. and W.A. Gale, "A Comparison of the Enhanced Good-Turing and Deleted Estimation Methods for Estimating Probabilities of English Bigrams," *Computer Speech and Language*, 1991, pp. 19-54.

[18] Cole, R., *et al.*, *Survey of the State of the Art in Human Language Technology*, eds. http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html, 1996, Cambridge University Press.

[19] Collins, M., "A New Statistical Parser Based on Bigram Lexical Dependencies," *ACL-96*, 1996, pp. 184-191.

[20] Darroch, J.N. and D. Ratcliff, "Generalized Iterative Scaling for Log-Linear Models," *The Annals of Mathematical Statistics*, 1972, **43**(5), pp. 1470-1480.

[21] Earley, J., *An Efficient Context-Free Parsing Algorithm*, PhD Thesis, 1968, Carnegie Mellon University, Pittsburgh.

[22] Earley, J., "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM*, 1970, 6(8), pp. 451-455.

[23] El-Bèze, M. and A.-M. Derouault, "A Morphological Model for Large Vocabulary Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1990, Albuquerque, NM, pp. 577-580.

[24] Fujisaki, T., *et al.*, "A probabilistic parsing method for sentence disambiguation," *Proc. of the Int. Workshop on Parsing Technologies*, 1989, Pittsburgh.

[25] Galescu, L., E.K. Ringger, and A.F. Allen, "Rapid Language Model Development for New Task Domains," *Proc. of the ELRA First Int. Conf. on Language Resources and Evaluation (LREC)*, 1998, Granada, Spain.

[26] Gazdar, G., *et al.*, *Generalized Phrase Structure Grammars*, 1985, Cambridge, MA, Harvard University Press.

[27] Gillett, J. and W. Ward, "A Language Model Combining Trigrams and Stochastic Context-Free Grammars," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[28] Good, I.J., "The Population Frequencies of Species and the Estimation of Population Parameters," *Biometrika*, 1953, pp. 237-264.

[29] Goodman, J., *Parsing Inside-Out*, PhD Thesis in Computer Science, 1998, Harvard University, Cambridge.

[30] Graham, S.L., M.A. Harrison, and W. L.Ruzzo, "An Improved Context-Free Recognizer," *ACM Trans. on Programming Languages and Systems*, 1980, 2(3), pp. 415-462.

[31] Hindle, D. and M. Rooth, "Structural Ambiguity and Lexical Relations," *DARPA Speech and Natural Language Workshop*, 1990, Hidden Valley, PA, Morgan Kaufmann.

[32] Hopcroft, J.E. and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 1979, Reading, MA, Addision Wesley.

[33] Iyer, R., M. Ostendorf, and J.R. Rohlicek, "Language Modeling with Sentence-Level Mixtures," *Proc. of the ARPA Human Language Technology Workshop*, 1994, Plainsboro, NJ, pp. 82-86.

[34] Jardino, M., "Multilingual Stochastic *N*-gram Class Language Models," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1996, Atlanta, GA, pp. 161-163.

[35] Jelinek, F., "Up From Trigrams! The Struggle for Improved Language Models" in *Proc. of the European Conf. on Speech Communication and Technology*, 1991, Genoa, Italy, pp. 1037-1040.

[36] Jelinek, F., *Statistical Methods for Speech Recognition*, 1998, Cambridge, MA, MIT Press.

[37] Jelinek, F., *et al.*, "A dynamic language model for speech recognition" in *Proc. of the DARPA Speech and Natural Language Workshop*, 1991, Asilomar, CA.

[38] Katz, S.M., "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *IEEE Trans. Acoustics, Speech and Signal Processing*, 1987(3), pp. 400-401.

[39] Kneser, R., "Statistical Language Modeling using a Variable Context" in *Proc. of the Int. Conf. on Spoken Language Processing*, 1996, Philadelphia, PA, p. 494.

[40] Kneser, R. and H. Ney, "Improved Backing-off for N-gram Language Modeling" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1995, Detroit, MI, pp. 181-184.

[41] Kuhn, R. and R.D. Mori, "A Cache-Based Natural Language Model for Speech Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1990(6), pp. 570-582.

[42] Lafferty, J.D. and B. Suhm, "Cluster Expansions and Iterative Scaling for Maximum Entropy Language Models" in *Maximum Entropy and Bayesian Methods*, K. Hanson and R. Silver, eds., 1995, Kluwer Academic Publishers.

[43] Lari, K. and S.J. Young, "Applications of Stochastic Context-free Grammars Using the Inside-Outside Algorithm," *Computer Speech and Language*, 1991, 5(3), pp. 237-257.

[44] Lau, R., R. Rosenfeld, and S. Roukos, "Trigger-Based Language Models: A Maximum Entropy Approach," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1993, Minneapolis, MN, pp. 108-113.

[45] Magerman, D.M. and M.P. Marcus, "Pearl: A Probabilistic Chart Parser," *Proc. of the Fourth DARPA Speech and Natural Language Workshop*, 1991, Pacific Grove, California.

[46] Mahajan, M., D. Beeferman, and X.D. Huang, "Improved Topic-Dependent Language Modeling Using Information Retrieval Techniques," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1999, Phoenix, AZ, pp. 541-544.

[47] Maltese, G. and F. Mancini, "An Automatic Technique to Include Grammatical and Morphological Information in a Trigram-based Statistical Language Model," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1992, San Francisco, CA, pp. 157-160.

[48] Moisa, L. and E. Giachin, "Automatic Clustering of Words for Probabilistic Language Models" in *Proc. of the European Conf. on Speech Communication and Technology* 1995, Madrid, Spain, pp. 1249-1252.

[49] Moore, R., *et al.*, "Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS," *Proc. of the ARPA Spoken Language Sys-*

tems Technology Workshop, 1995, Austin, Texas, Morgan Kaufmann, Los Altos, CA.

[50] Nasr, A., *et al.*, "A Language Model Combining *N*-grams and Stochastic Finitie State Automata," *Proc. of the Eurospeech*, 1999, Budapest, Hungary, pp. 2175-2178.

[51] Pereira, F.C.N. and Y. Schabes, "Inside-Outside Reestimation from Partially Bracketed Corpora," *Proc. of the 30th Annual Meeting of the Association for Computational Linguistics*, 1992, pp. 128-135.

[52] Pietra, S.A.D., *et al.*, "Adaptive Language Model Estimation using Minimum Discrimination Estimation," *Proc. of the IEEE Int. Conf. on Acoustics. Speech and Signal Processing*, 1992, San Francisco, CA, pp. 633-636.

[53] Pollard, C. and I.A. Sag, *Head-Driven Phrase Structure Grammar*, 1994, Chicago, University of Chicago Press.

[54] Pullum, G. and G. Gazdar, "Natural Languages and Context-Free Languages," *Linguistics and Philosophy*, 1982, **4**, pp. 471-504.

[55] Ratnaparkhi, A., S. Roukos, and R.T. Ward, "A Maximum Entropy Model for Parsing," *Proc. of the Int. Conf. on Spoken Language Processing*, 1994, Yokohama, Japan, pp. 803-806.

[56] Rosenfeld, R., *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, Ph.D. Thesis in School of Computer Science, 1994, Carnegie Mellon University, Pittsburgh, PA.

[57] Salton, G. and M.J. McGill, *Introduction to Modern Information Retrieval*, 1983, New York, McGraw-Hill.

[58] Schabes, Y., M. Roth, and R. Osborne, "Parsing the *Wall Street Journal* with the Inside-Outside Algorithm," *Proc. of the Sixth Conf. of the European Chapter of the Association for Computational Linguistics*, 1993, pp. 341-347.

[59] Seymore, K. and R. Rosenfeld, "Scalable Backoff Language Models," *Proc. of the Int. Conf. on Spoken Language Processing*, 1996, Philadelphia, PA, pp. 232.

[60] Shannon, C.E., "Prediction and Entropy of Printed English," *Bell System Technical Journal*, 1951, pp. 50-62.

[61] Sharman, R., F. Jelinek, and R.L. Mercer, "Generating a Grammar for Statistical Training," *Proc. of the Third DARPA Speech and Natural Language Workshop*, 1990, Hidden Valley, Pennsylvania, pp. 267-274.

[62] Shieber, S.M., *An Introduction to Unification-Based Approaches to Grammars*, 1986, Cambridge, UK, CSLI Publication, Leland Stanford Junior University.

[63] Steinbiss, V., *et al.*, "A 10,000-word Continuous Speech Recognition System," *Proc. of the IEEE Int. Conf. on Acoustics. Speech and Signal Processing*, 1990, Albuquerque, NM, pp. 57-60.

[64] Stolcke, A., "Entropy-based Pruning of Backoff Language Models," *DARPA Broadcast News Transcription and Understanding Workshop*, 1998, Lansdowne, VA.

[65] Tomita, M., "An Efficient Augmented-Context-Free Parsing Algorithm," *Computational Linguistics*, 1987, **13**(1-2), pp. 31-46.

[66] Wang, Y., M. Mahajan, and X. Huang, "A Unified Context-Free Grammar and *N*-Gram Model for Spoken Language Processing," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey, pp. 1639-1642.

[67] Younger, D.H., "Recognition and Parsing of Context-Free Languages in Time n^3," *Information and Control*, 1967, **10**, pp. 189-208.

CHAPTER 12

Basic Search Algorithms

Continuous speech recognition (CSR) is both a pattern recognition and search problem. As described in previous chapters, the acoustic and language models are built upon a statistical pattern recognition framework. In speech recognition, making a search decision is also referred to as decoding. In fact, decoding got its name from information theory (see Chapter 3) where the idea is to *decode* a signal that has presumably been encoded by the source process and has been transmitted through the communication channel, as depicted in Chapter 1, Figure 1.1. In this chapter, we first review the general decoder architecture that is based on such a source-channel model.

The decoding process of a speech recognizer is to find a sequence of words whose corresponding acoustic and language models best match the input signal. Therefore, the process of such a decoding process with trained acoustic and language models is often referred to as just a *search* process. Graph search algorithms have been explored extensively in the fields of artificial intelligence, operation research, and game theory. In this chapter first we present several basic search algorithms, which serve as the basic foundation for CSR.

The complexity of a search algorithm is highly correlated with the search space, which is determined by the constraints imposed by the language models. We discuss the impact of different language models, including finite-state grammars, context-free grammars, and n-grams.

Speech recognition search is usually done with the Viterbi or A* stack decoders. The reasons for choosing the Viterbi decoder involve arguments that point to speech as a left-to-right process and to the efficiencies afforded by a time-synchronous process. The reasons for choosing a stack decoder involve its ability to more effectively exploit the A* criteria, which holds out the hope of performing an optimal search as well as the ability to handle huge search spaces. Both algorithms have been successfully applied to various speech recognition systems. The relative merits of both search algorithms were quite controversial in the 1980s. Lately, with the help of efficient pruning techniques, Viterbi beam search has been the preferred method for almost all speech recognition tasks. Stack decoding, on the other hand, remains an important strategy to uncover the n-best and lattice structures.

12.1. BASIC SEARCH ALGORITHMS

Search is a subject of interest in artificial intelligence and has been well studied for expert systems, game playing, and information retrieval. We discuss several general graph search methods that are fundamental to spoken language systems. Although the basic concept of graph search algorithms is independent of any specific task, the efficiency often depends on how we exploit domain-specific knowledge.

The idea of search implies moving around, examining things, and making decisions about whether the sought object has yet been found. In general, search problems can be represented using the *state-space search* paradigm. It is defined by a triplet (S, O, G), where S is a set of initial states, O a set of operators (or rules) applied on a state to generate a transition with its corresponding cost to another state, and G a set of goal states. A solution in the state-space search paradigm consists in finding a path from an initial state to a goal state. The state-space representation is commonly identified with a directed graph in which each node corresponds to a state and each arc to an application of an operator (or a rule), which transitions from one state to another. Thus, the state-space search is equivalent to searching through the graph with some objective function.

Before we present any graph search algorithms, we need to remind the readers of the importance of the dynamic programming algorithm described in Chapter 8. Dynamic programming should be applied whenever possible and as early as possible because (1) unlike any heuristics, it will not sacrifice optimality; (2) it can transform an exponential search into a polynomial search.

12.1.1. General Graph Searching Procedures

Although dynamic programming is a powerful polynomial search algorithm, many interesting problems cannot be handled by it. A classical example is the traveling salesman's problem. We need to find a shortest-distance tour, starting at one of many cities, visiting each city exactly once, and returning to the starting city. This is one of the most famous problems in the *NP*-hard class [1, 32]. Another classical example is the *N*-queens problem (typically 8-queens), where the goal is to place N queens on an $N \times N$ chessboard in such a way that no queen can capture any other queen, i.e., there is no more than one queen in any given row, column, or diagonal. Many of these puzzles have the same characteristics. As we know, the best algorithms currently known for solving the *NP*-hard problem are exponential in the problem size. Most graph search algorithms try to solve those problems using heuristics to avoid or moderate such a combinatorial explosion.

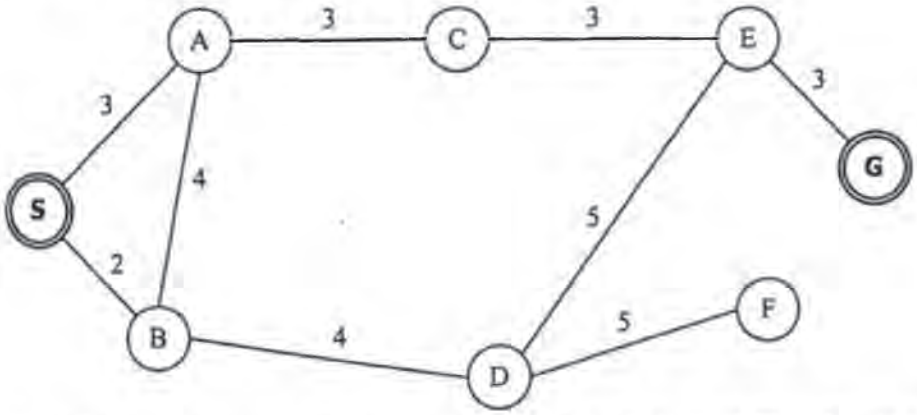


Figure 12.1 A highway distance map for cities S, A, B, C, D, E, F, and G. The salesman needs to find a path to travel from city S to city G [42].

Let's start our discussion of graph search procedure with a simple city-traveling problem [42]. Figure 12.1 shows a highway distance map for all the cities. A salesman named John needs to travel from the starting city S to the end city G. One obvious way to find a path is to derive a graph that allows orderly exploration of all possible paths. Figure 12.2 shows the graph that traces out all possible paths in the city-distance map shown in Figure 12.1. Although the city-city connection is bi-directional, we should note that the search graph in this case must not contain cyclic paths, because they would not lead to any progress in this scenario.

If we define the search space as the potential number of nodes (states) in the graph search procedure, the search space for finding the optimal state sequence in the Viterbi algorithm (described in Chapter 8) is $N \times T$, where N is the number of states for the HMM and T is the length of the observation. Similary, the search space for John's traveling problem will be 27.

Another important measure for a search graph is the *branching factor*, defined as the average number of successors for each node. Since the number of nodes of a search graph

(or tree) grows exponentially with base equal to this branching factor, we certainly need to watch out for search graphs (or trees) with a large branching factor. Sometimes they can be too big to handle (even infinite, as in game playing). We often trade the optimal solution for improved performance and feasibility. That is, the goal for such search problems is to find one satisfactory solution instead of the optimal one. In fact, most AI (artifical intelligence) search problems belong to this category.

The search tree in Figure 12.2 may be implemented either explicitly or implicitly. In an explicit implementation, the nodes and arcs with their corresponding distances (or costs) are explicitly specified by a table. However, an explicit implementation is clearly impractical for large search graphs and impossible for those with infinite nodes. In practice, most parts of the graph may never be explored before a solution is found. Therefore, a sensible strategy is to dynamically generate the search graph. The part that becomes explicit is often referred to as an *active* search space. Throughout the discussion here, it is important to keep in mind this distinction between the implicit search graph that is specified by the start node S and the explicit partial search graphs that are actually constructed by the search algorithm.

To expand the tree, the term *successor operator* (or *move generator*, as it is often called in game search) is defined as an operator that is applied to a node to generate all of the successors of that node and to compute the distance associated with each arc. The successor operator obviously depends on the topology (or rules) of the problem space. Expanding the starting node S, and successors of S, ad infinitum, gradually makes the implicitly

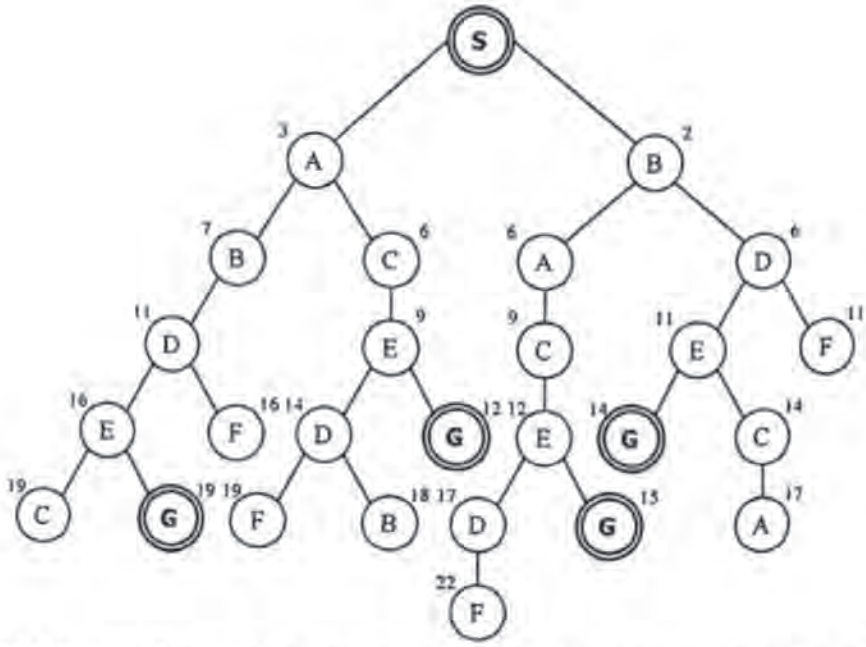


Figure 12.2 The search tree (graph) for the salesman problem illustrated in Figure 12.1. The number next to each node is the accumulated distance from start city to end city [42].