system can be used to provide a good alignment upon which to base the derivation of a decision tree for a word-boundary-dependent quinphone system.

As well as incorporating context dependency in the acoustic models, it can be beneficial to deal with male and female speech separately, because the differences between male and female speech tend to be much greater than the differences between talkers of the same sex. Models built in this way are commonly described as **gender dependent**. One way of training gender-dependent models robustly is to introduce gender dependency at the final training stage by cloning the trained multiple-component mixture models and then re-estimating the state means and mixture weights for each model set using the data for the male and female speech separately. State variances are usually kept gender independent in order to avoid the robustness issues that would otherwise arise due to the more limited data available to train variances for each gender separately. Once the gender-dependent models have been trained, a straightforward method for using them in recognition is to run two recognizers in parallel, one using the 'male' models and the other using the 'female' models. The system output is then taken from the recognizer which gives the highest probability for an utterance (where the utterance represents a sequence of words known to have been spoken by the same person).

12.7 LANGUAGE MODELLING

In any language, there are syntactic, semantic and pragmatic constraints that have the effect of making some sequences of words more likely than others. For an ASR system that is intended for a particular application domain, the language that the system can recognize may be quite limited, in terms both of vocabulary size and of utterance syntax. However, for more general recognition of large vocabularies, *any* word sequence must be allowed, but different probabilities need to be assigned to different sequences. The purpose of the language model is to make effective use of linguistic constraints when computing the probability of the different possible word sequences. Assuming a sequence of K words, $W = w_1, w_2, \ldots, w_K$, the sequence probability $P(W)$ can be expanded in terms of conditional probabilities as follows:

$$P(W) = P(w_1, w_2, \ldots, w_K) = \prod_{k=1}^{K} P(w_k \mid w_1, \ldots, w_{k-1}) \,. \tag{12.3}$$

This expression simply states that the probability of the word sequence can be decomposed into the probability of the first word, multiplied by the probability of the second word given the first word, multiplied by the probability of the third word given the first and the second words, and so on for all the words in the sequence. Initially, the probabilities of the words will be influenced mostly by general constraints on the types of words that are most likely to start utterances. As the utterance continues, an increasing number of words will have already been spoken and so it becomes easier to predict the next word.

For any natural language, there is of course a vast number of possible word combinations and hence a huge number of conditional probabilities that might be required. It is not feasible to specify all of these probabilities individually, and so some modelling assumptions are needed to make the task of specifying the conditional probabilities more manageable.

12.7.1 *N*-grams

A simple solution to the problem of estimating word-sequence probabilities is to use ***N*-grams** (where N is a small number). Here it is assumed that the probability of observing any word w_k depends only on the identity of the previous $N-1$ words (so that each conditional probability depends on a total of N words, including the current word). Using N-grams Equation (12.3) can be approximated as follows:

$$P(W) = \prod_{k=1}^{K} P(w_k \mid w_1, \ldots, w_{k-1}) \approx \prod_{k=1}^{K} P(w_k \mid w_{k-N+1}, \ldots, w_{k-1}). \tag{12.4}$$

If $N=1$ the model is a **unigram** and just represents the probability of the word. A **bigram** ($N=2$) models the probability of a word given the immediately preceding word, while a **trigram** ($N=3$) takes into account the previous two words.

To illustrate the use of N-grams to estimate word-sequence probabilities, consider the phrase "ten pots fell over". For a unigram model, the probability of the sequence is obtained simply by multiplying together the word probabilities:

$$P(\text{ten pots fell over}) \approx P(\text{ten})\, P(\text{pots})\, P(\text{fell})\, P(\text{over}). \tag{12.5}$$

In the case of a bigram model the probability of the sequence is estimated as:

$$\begin{aligned} P(\text{ten pots fell over}) \approx\ & P(\text{ten} \mid \text{START})\, P(\text{pots} \mid \text{ten}) \\ & P(\text{fell} \mid \text{pots})\, P(\text{over} \mid \text{fell}), \end{aligned} \tag{12.6}$$

where START is used to indicate the beginning of the sequence. For a trigram model, the probability estimate becomes:

$$\begin{aligned} P(\text{ten pots fell over}) \approx\ & P(\text{ten} \mid \text{START})\, P(\text{pots} \mid \text{START ten}) \\ & P(\text{fell} \mid \text{ten pots})\, P(\text{over} \mid \text{pots fell}). \end{aligned} \tag{12.7}$$

In principle, N-grams can be estimated using simple frequency counts from training data to obtain maximum-likelihood estimates for the required probabilities. Considering the bigram (w_{k-1}, w_k) and the trigram (w_{k-2}, w_{k-1}, w_k), the conditional probabilities $P(w_k \mid w_{k-1})$ and $P(w_k \mid w_{k-2}, w_{k-1})$ could be estimated thus:

$$\hat{P}(w_k \mid w_{k-1}) = \frac{C(w_{k-1}, w_k)}{C(w_{k-1})}, \quad \hat{P}(w_k \mid w_{k-2}, w_{k-1}) = \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})}, \tag{12.8}$$

where the notation $C(x)$ is used to represent the count of number of examples of x. For the examples of the bigram (fell over) and the trigram (pots fell over) we have:

$$\hat{P}(\text{over} \mid \text{fell}) = \frac{C(\text{fell over})}{C(\text{fell})}, \quad \hat{P}(\text{over} \mid \text{pots fell}) = \frac{C(\text{pots fell over})}{C(\text{fell over})}. \tag{12.9}$$

12.7.2 Perplexity and evaluating language models

The task of the language model can be viewed as one of predicting words in a sequence, and a good language model will be one that provides a good predictor of

the word in any position based on the words observed so far. Given a particular sequence of K words in some test database, the value of $P(W)$ for that sequence provides a measure of how well the language model can predict the sequence: the higher the value of $P(W)$, the better the language model is at predicting the word sequence. Word sequences differ in length, and an average measure of the probability per word is obtained by taking the Kth root of the probability of the sequence. The inverse of this probability defines the **perplexity**, $PP(W)$, thus:

$$PP(W) = \left[P(w_1, w_2, \ldots, w_K)\right]^{-1/K} = \left[\prod_{k=1}^{K} P(w_k \mid w_1, \ldots, w_{k-1})\right]^{-1/K}. \qquad (12.10)$$

For any given language model, it is possible to calculate the perplexity of some corpus to be recognized. For artificially constrained tasks with a rigid syntax, perplexity can be interpreted as being equivalent to the average number of different words that would need to be distinguished at any point in the word sequence, if all words at any particular point were equiprobable. Perplexity is therefore sometimes referred to as representing an **average branching factor**. The lower the value of perplexity, the fewer the number of alternatives that must be considered. The lowest possible value of perplexity is 1, but this value would only be obtained if all the individual word probabilities were equal to 1, such that only one word sequence could be recognized by the system. At the other extreme, if any word in a sequence is assigned a probability of zero by the language model, then the probability of the complete sequence will be equal to zero and the perplexity will be infinitely large.

As we have already seen, a major challenge for any model of natural language is to avoid probabilities of zero by not excluding any of the vocabulary words, while making the prediction of the next word as good as possible by having only a few high-probability alternatives at any one point. A good language model should give a low value of perplexity on a large corpus of representative text material (with no part of that material having previously been used to train the model).

The perplexity measure provides a means for evaluating alternative possible language models on some test corpus without needing to run a complete recognition experiment. A perplexity evaluation allows the language-model component to be assessed independently from the acoustic-model component, but it cannot take into account any interactions between the two models: good discrimination by the language model may not have much effect on recognition performance if the words concerned are acoustically very distinct, but could have a large effect for acoustically confusable words. Furthermore, any effects of the search algorithm cannot be allowed for in the perplexity calculation. Thus perplexity on a test data set is helpful for comparing alternative language models and also provides a useful indicator of the difficulty of the recognition task to be performed by the acoustic models, but the final test must be in terms of the recognition accuracy of the complete system.

12.7.3 Data sparsity in language modelling

Maximum-likelihood estimates obtained from frequency counts using expressions such as those in Equation (12.8) are a good approximation to the true probabilities,

provided that the sample size is large in relation to the number of possible outcomes. Unfortunately, in the case of N-gram modelling there are a vast number of possible outcomes. A vocabulary of V words provides V^2 potential bigrams and V^3 potential trigrams. For a 20,000-word vocabulary there are thus 400 million possible word bigrams and eight million million possible trigrams! It would be completely impracticable to provide sufficient speech data to determine the language-model probabilities, and instead very large text corpora are used. Even so, while typical text corpora may contain over 100 million words, most of the possible bigrams and the vast majority of possible trigrams will not occur at all in the training data and many others will only appear once or twice.

Data sparsity is a much greater problem for the language model than for the acoustic model, due to the much larger size of the inventory of basic units (words rather than phones). As a consequence of this severe data-sparsity problem, it is not possible to rely only on simple frequency counts for estimating language-model probabilities, as many of the bigram and trigram probabilities would be set to zero (so that it would then be impossible to recognize any word sequence containing one of these combinations) and many other probabilities would be poorly estimated. The successful use of N-grams for LVCSR is dependent upon the use of **smoothing** techniques for obtaining accurate, robust (non-zero) probability estimates for *all* the possible N-grams that can occur for a given vocabulary. Zero probabilities and low non-zero probabilities are adjusted upwards, and high probabilities are reduced. Thus the overall effect is to make the probability distributions more uniform, and hence 'smoother'. Some different aspects of smoothing algorithms are described briefly in the following sections.

12.7.4 Discounting

For any set of possible 'events', such as bigrams or trigrams, the sum of the probabilities for all the possibilities must be equal to one. When only a subset of the possible events occurs in the training data, the sum of the probabilities of all the observed events must therefore be less than one. This effect can be achieved by reducing the observed frequency counts. The process is generally known as **discounting**, and is often described in terms of 'freeing' **probability mass** from the observed events which can then be redistributed among the unseen events.

Several different methods have been used to achieve discounting, and these methods differ in the way in which the reduced probabilities are calculated. Full coverage of discounting methods is outside the scope of this book but, for example, one simple but effective technique is **absolute discounting**. Here some small fixed amount (between zero and one) is subtracted from each frequency count (the numerators in Equation (12.8) or in the examples shown in Equation (12.9)). Thus the probability of every observed event is reduced, but the effect decreases as the observed frequency count increases (when the maximum-likelihood estimate should be more reliable). However, the same discount value may not be optimum for the full range of frequency counts. In particular, there is some evidence that absolute discounting imposes too great a reduction in the probability of events that occur only once or twice. A variant of the technique overcomes this problem by having separate discount values for these rare events.

12.7.5 Backing off in language modelling

Probability mass which has been made available as a result of discounting can be allocated to the unseen events. In estimating the probabilities of the unseen events, it is desirable to make use of any information that is available about the relative probabilities of the different events. In the case of N-grams, useful information may be available in the form of the probability according to a more general distribution. Thus it is possible to formulate a recursive procedure of '**backing off**': if a trigram is not observed, the model backs off to the relevant bigram probability, and if the bigram is not available, it backs off further to the unigram probability. For any words which do not occur in the training texts at all, it is possible to back off to a uniform distribution whereby all words are assumed to be equally likely.

Backing off can be illustrated by considering the task of estimating the conditional trigram probabilities $P(w_k | w_{k-2}, w_{k-1})$ for word w_k in the context of the sequence of preceding words (w_{k-2}, w_{k-1}). We will assume that some appropriate discounting method has been used to assign probabilities to all observed trigrams, and these probability estimates will be denoted $\widetilde{P}(w_k | w_{k-2}, w_{k-1})$. Thus, using $P_s(w_k | w_{k-2}, w_{k-1})$ to denote an estimate for the probability of word w_k given the sequence of preceding words (w_{k-2}, w_{k-1}), a backing-off scheme for obtaining this probability estimate is as follows:

$$P_s(w_k | w_{k-2}, w_{k-1}) = \begin{cases} \widetilde{P}(w_k | w_{k-2}, w_{k-1}) & \text{if } C(w_{k-2}, w_{k-1}, w_k) > 0 \\ B(w_{k-2}, w_{k-1})\, P_s(w_k | w_{k-1}) & \text{otherwise.} \end{cases} \tag{12.11}$$

$P_s(w_k | w_{k-1})$ is the estimated bigram probability of w_k given preceding word w_{k-1}. $B(w_{k-2}, w_{k-1})$ is a normalizing constant for trigram context (w_{k-2}, w_{k-1}), and scales the bigram probabilities so that the sum of the $P_s(w_k | w_{k-2}, w_{k-1})$ terms is equal to 1 when computed over all possible words w_k. The sum of all the probabilities that are calculated by backing off must be equal to the total probability mass that has been freed from discounting the relevant trigram context. The normalizing constant is therefore chosen to be equal to the appropriate fraction of this freed probability.

Backing off from bigram to unigram and from unigram to uniform distributions can be accomplished in an analogous manner.

By setting the count threshold to zero, backing off only occurs for N-grams with no examples at all. However, as observing just one or two examples is unlikely to provide a reliable probability estimate, it can be beneficial to apply a higher threshold and so disregard those N-grams that occur just a few times in the training data. In this way only robustly estimated N-grams should be retained, which has the added benefit of a substantial saving in the memory needed for the language model.

12.7.6 Interpolation of language models

Backing off involves choosing between a specific and a more general distribution. An alternative is to compute a weighted average of different probability estimates, obtained for contexts that can range from very specific to very general. The idea is to improve the accuracy and robustness of a context-specific probability estimate by combining it with more general estimates for which more data are available. One option involves taking a linear combination of different probability estimates. For

example, a trigram probability could be estimated by linear interpolation between relative frequencies of the relevant trigrams, bigrams and unigrams, thus:

$$P_s(w_k \mid w_{k-2}, w_{k-1}) = \lambda_3 \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})} + \lambda_2 \frac{C(w_{k-1}, w_k)}{C(w_{k-1})} + \lambda_1 \frac{C(w_k)}{K}, \quad (12.12)$$

where K is the number of different words, and the sum of the non-negative weights $\lambda_1 + \lambda_2 + \lambda_3 = 1$. The values for the weights need to be chosen to make the best compromise between specificity and ability to generalize to new data. The training data are therefore divided into two parts. The first (larger) part of the data is used to derive the frequency counts, which are then used when finding the optimum values of the weights in order to maximize the probability for the second part of the data. Because the parameters that are estimated will tend to depend on how the data are partitioned, often alternative sets of parameters are estimated for several different ways of splitting the data, and then the individual estimates are combined. This smoothing method is often called **deleted interpolation**.

For simplicity, interpolation has been introduced using probability estimates obtained from simple frequency counts. However, smoothing by interpolation can also be applied to other probability estimates, including N-gram probabilities that have been obtained by, for example, the absolute discounting technique mentioned in Section 12.7.4. Interpolation schemes can also be used with probabilities from other types of language model, such as those discussed in Section 12.7.8 below.

12.7.7 Choice of more general distribution for smoothing

In the previous sections we have described ways of smoothing N-grams using lower-order N-grams. The lower-order distribution is usually defined in a way which is exactly analagous to the definition for the higher-order distribution that is being smoothed. There is however an alternative method which may give more reliable estimates of the lower-order probabilities. This method is best explained by describing an example. Consider a word, such as "Francisco", that almost always occurs following just one other word ("San"). If the training text contains several examples of "San Francisco", both the bigram probability P(Francisco|San) and the unigram probability P(Francisco) will be high. Thus if we then use a discounting method such as absolute discounting to derive the probability of "Francisco" occurring after some other bigram history, this probability will also tend to be fairly high. However, intuitively it seems more plausible that, if the only information we have implies that "Francisco" always follows "San", the probability of "Francisco" following any other word should be low.

It is possible to define the unigram probability used for smoothing not in terms of number of examples of a word but rather in terms of the number of *different* contexts in which the word occurs. Chen and Goodman (1999) have proposed using this approach together with smoothing by interpolation, and incorporating a variant of absolute discounting that uses three separate discounts (one for N-grams occurring just once, one for N-grams occurring twice, and a third for all other N-gram counts). In comparative experiments, they demonstrated that this method performed consistently better (in terms of both perplexity and recognition performance) than a wide range of other smoothing techniques.

12.7.8 Improving on simple *N*-grams

N-grams have proved to be a very popular approach to language modelling. Bigrams and trigrams are the most widely used but 4-grams and even 5-grams are sometimes also included. *N*-grams provide a simple representation of language structure by focusing on local dependencies based only on word identity. Effects due to syntax, semantics and pragmatics are captured simultaneously but with no distinction between them. Although the method is really very crude, in practice it is very effective for languages such as English for which word order is important and the strongest contextual effects tend to be from immediately adjacent words.

While *N*-grams are good at modelling local context, an obvious deficiency is their inability to capture longer-term effects: syntactic constraints (e.g. subject–verb agreement) and semantic influences (certain words tending to occur together) may both operate over a span of several words. Various methods have been suggested to incorporate these effects, usually to provide additional information that can be interpolated with *N*-gram probabilities. A few of the developments are briefly mentioned in the following paragraphs.

N-grams are simple to compute directly from text data, without any need for explicit linguistic knowledge about the individual words. However, if information about syntactic classes (or other word groupings) is available, it is possible to estimate class-based *N*-gram probabilities and to back off or interpolate using these rather than needing to go to a shorter context (e.g. using a class-based trigram rather than resorting to a bigram). Taking this idea further, decision-tree methods can be applied to language models to find the best way of partitioning the data into different clusters based on questions about syntactic and/or semantic context. Other methods have involved attempting some grammatical analysis to determine the syntactic structure of the hypothesized sentence so far.

For generality, a language model needs to be trained on a large body of text from diverse sources. However, language is actually very dynamic in character, with the probability of many words being very different depending on the subject matter. One way of taking account of short-term patterns of word usage is to introduce a cache component (by analogy with "cache memory" in hardware terminology). The cache is simply a buffer which stores the frequency of occurrence of some number of the most recent words (typically around 200 different words). Word probabilities can be estimated by interpolating conventional *N*-gram probabilities with the probabilities as given by frequency of occurrence in the cache. A related concept is the idea of word 'triggers', whereby certain words tend to be very strong indicators, or triggers, for other words to occur in the general vicinity, but not necessarily within the span covered by a trigram. Triggers may be related to subject matter (e.g. "airline" associated with "flights") or to linguistic constructs (e.g. "neither" tends to be followed fairly soon afterwards by "nor"). Trigger models are used in conjunction with a cache component to keep a record of the recent words from which to adjust the probabilities for likely 'triggered' words.

The use of a cache and the incorporation of triggers are two ways of capturing dynamics in language usage. It is also possible to apply language-model adaptation techniques (analogous to the acoustic-model adaptation methods described in Chapter 11) to adapt *N*-gram probabilities based on adaptation data, which might for example relate to a new topic or reflect talker-specific language patterns.

12.8 DECODING

The recognition task is to find the most probable sequence of words $\hat{W}$, as given by Equation (12.1). As the vocabulary size becomes larger, the number of different possible word sequences soon becomes prohibitive and any practical system needs an efficient means of searching to find the most probable sequence. The component of the system that performs this search operation is often referred to as the **decoder**.

Although recognition using whole-word HMMs is not practicable with a large vocabulary, the same principles can be applied to units of any size. In the previous sections we have explained how statistical models can be used at different levels. One level captures the relationship between phone-size units and their acoustic realization, then there are the different phone sequences that can make up a word, and finally there are the probabilities of different word sequences. Thus a language can be modelled as a network of states representing linguistic structure at many levels. At the lowest level a small network of states represents a triphone or similar unit. At the next level a network of triphones forms a state to represent a word. A complete sentence can then be generated by a network of word states, where the connections between the states correspond to the language-model probabilities. The decoding task is to find the best path through this multiple-level network, and the recognized word sequence is given by the best path at the highest level.

In order to apply the principles of DP using the Viterbi algorithm to a multiple-level network, probabilities need to be evaluated for all valid partial paths at all levels, with no decisions being reached until all earlier parts of a path enter the same highest-level state at the same time, thus delaying decisions until all relevant evidence has been used. The use of delayed decisions is a fundamental principle of this HMM approach, as it enables the knowledge and constraints at all levels to be employed simultaneously to find the best explanation of the data. In practice, even at any one point in time, in a large-vocabulary system there are so many possibilities (both at the word and at the sub-word level) that it is impossible to evaluate all of them. However, the language model provides strong constraints that act to make many of the possibilities extremely unlikely, and it is necessary to find an efficient way of applying these constraints within the decoding.

In a first-order HMM the probability of a transition from any one state to any other state depends only on the identities of the source and destination states. This model cannot accommodate a trigram (or higher-order) language model, because here the word transition probability depends on more than just the immediately preceding state. In order to use such a language model, some way of dealing with the higher-order dependencies needs to be found. An additional complication arises when using cross-word triphones because the identity of a word-final triphone, and hence the word probability, depends on the identity of the following word.

The issues associated with large-vocabulary decoding have been addressed in various different ways. Three main types of method are briefly described below.

12.8.1 Efficient one-pass Viterbi decoding for large vocabularies

In order to accommodate cross-word triphone models, the state network for a Viterbi search needs to include multiple entries for each word to cover all possible

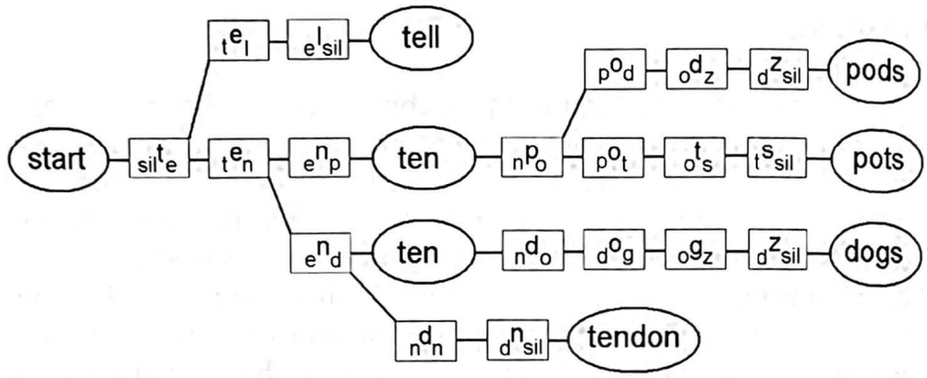


Figure 12.4 A very small fragment of a network for decoding using cross-word triphone models. This fragment shows the sequence "ten pots" and a few possible alternatives that branch at different positions in the network. Note that the word "ten" is represented by different nodes in the network, depending on whether the following word is "pots" or "dogs".

different triphones that may end any one word. Similarly, a trigram language model can be used by expanding the network to keep multiple copies of each word so that each transition between words has a unique two-word history. To make the network manageable, it is usually represented as a tree structure, as shown in Figure 12.4. In the tree, different hypotheses that start with the same sequence of sub-word models share those models. This tree network is built dynamically as required.

In Section 8.8 we introduced the concept of score pruning to reduce the number of hypotheses to be evaluated in a DP search. Efficient pruning is essential for LVCSR systems. The usual scheme employs a **beam search**, whereby at each time frame all paths whose likelihood score is not within some specified threshold of the best-scoring path are pruned out. In practice, it is generally the case that the likelihoods for all except just the few most likely states tend to be very small, so it is possible to concentrate the search on a narrow beam of possible alternatives.

Language constraints act to restrict the set of words that are likely at any given point in an utterance. It is therefore advantageous to use the language model to prune out unlikely possibilities as soon as possible when decoding an utterance, but in conventional Viterbi decoding the language-model probability is not known until the end of a word is reached. However, if a record is kept of the current possible words associated with each sub-word model, it is possible to use the language-model probabilities to prune out unlikely hypotheses at an earlier stage.

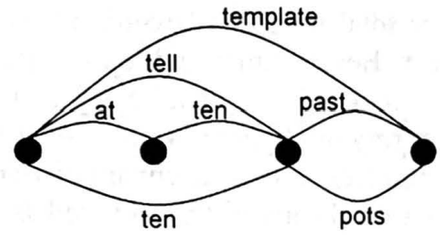
12.8.2 Multiple-pass Viterbi decoding

An alternative to attempting complete and accurate recognition in a single pass is to use a multiple-pass approach. The idea here is to start by using a simple recognition system to identify a small number of likely hypotheses, and then to use a more elaborate system to choose between these possibilities. For example, the first recognition pass could use only word-internal triphones and a bigram language model. A second recognition pass might then incorporate cross-word triphones and other more specific acoustic models as well as trigram and cache language models.

The output of the first recognition pass is usually expressed either as a rank-ordered ***N*-best** list of possible word sequences, or as a **word graph** or **lattice**

1 ten pots
2 tell pots
3 template
4 at ten past
5 ten past
6 tell past

(a) *N*-best list of hypotheses.



(b) Lattice representation of the alternatives shown in (a).

Figure 12.5 Possible alternative utterances that might be generated for a short utterance.

describing the possibilities as a network. Figure 12.5 shows an example of an *N*-best list and a word lattice that could be generated for an utterance of "ten pots". In order to output more than one possible sequence, the first-pass DP search needs to be extended to retain several hypotheses at each stage.

12.8.3 Depth-first decoding

The Viterbi approach is known as a **breadth-first** search, because all possibilities are considered in parallel, with all paths to time t being evaluated before proceeding to time $t + 1$. An alternative is to adopt a **depth-first** search, for which the principle is to pursue the most promising hypothesis until the end of the utterance. This strategy is often referred to as **stack decoding**: the idea is to keep an ordered 'stack' of possible hypotheses and to iteratively take the best hypothesis from the stack, choose the most likely next word and then add this hypothesis as a new entry to the stack, reordering the hypotheses as necessary. Once the end of the utterance is reached, the best-scoring hypothesis is output as the recognized word sequence. At any point in the search it may be necessary to compare hypotheses of different lengths. Because the score for any one path is a product of probabilities, it decreases with time and so a simple comparison of different scores will be biased towards shorter hypotheses. This problem can be overcome by normalizing the score for each path based on the number of frames that it accounts for.

With depth-first decoding a new hypothesis is generated each time a word is added to one of the existing hypotheses, so it is possible to evaluate the different options that are required for long-range language models. The method can, however, be expensive in terms of both memory and processing requirements.

12.9 EVALUATING LVCSR PERFORMANCE

12.9.1 Measuring errors

When recognizing connected speech, there are three types of recognition error: **substitution** (the wrong word is recognized), **deletion** (a word is omitted) and **insertion** (an extra word is recognized). The **word error rate (WER)** is given by:

$$\text{WER} = 100 \times \frac{C(\text{substitutions}) + C(\text{deletions}) + C(\text{insertions})}{N}\,\%, \qquad (12.13)$$

where N is the total number of words in the test speech and $C(x)$ is the count of errors of type x. Because there will not in general be a one-to-one correspondence between the recognized and actual word sequences, a DP alignment process is used to find the best way of aligning the two before the WER can be calculated.

The percentage **word accuracy** is equal to 100 – WER. Sometimes the percentage of words correctly recognized is quoted, but this measure may not be such a good indicator of true performance as it does not include insertion errors.

12.9.2 Controlling word insertion errors

The probability that is assigned to any word sequence will depend on the relative contributions from the language and acoustic models. However, both models involve approximations and assumptions, and hence only provide estimates of the relevant probabilities. In particular, the probability given by the acoustic model will depend on the number and choice of acoustic features that are used and typically has a disproportionately large influence relative to the language-model probability. When the language model is given insufficient weight, the consequence is often a large number of errors due to insertion of many short function words. The short duration and high variability that are typical of these words mean that a sequence of their models may provide the best acoustic match to short regions of speech, even though the word sequence is given a low probability by the language model.

There are two practical solutions that are often adopted to tackle the problem of word insertion errors. One approach is to impose a **word insertion penalty**, whereby the probability of making a transition from one word to another is explicitly penalized by multiplying the word probability by some number less than 1. Alternatively or in addition, the influence of the language model on the combined probability can be increased by raising its probability to some power greater than 1. Values, both for this power and for any word insertion penalty, are usually chosen experimentally to maximize performance on some evaluation data.

12.9.3 Performance evaluations

By the mid-1980s HMMs were showing promise for practical application to large-vocabulary recognition tasks. It was, however, difficult to compare competing systems from different laboratories because individual systems tended to be trained and tested on different sets of data, due to the absence of any common databases for training or for testing. This widely recognized problem was addressed when ARPA (also known as DARPA, the *Defense* Advanced Research Projects Agency) began funding a second major research programme in the area of ASR. As part of this programme, a recognition task was defined and speech data were recorded and made publicly available. The data were divided into a **training set** for training a set of models, a **development test set** for testing a recognition system during its development, and an **evaluation test set** on which the final system could be tested and scored just once. The advent of this type of database enabled direct comparisons to be made between different algorithms and systems, with controlled assessments of performance that could also be used to measure progress over time.

In 1989 the first formal competitive evaluation to measure the performance of different systems was organized. This evaluation marked the start of a series of tests that is still continuing today. Any organization participating in an evaluation has to test its recognition system on the designated evaluation test data (following the pre-specified rules of the evaluation) and then submit the recognition results for scoring by the National Institute of Standards and Technology (NIST), a U.S. government organization. This scoring of the results by an independent body helps to ensure the applicability of direct and fair comparisons between different systems.

All the recognition tasks that have been chosen by ARPA have involved speaker-independent recognition, but over the years the tasks have progressively become more challenging both by increasing the vocabulary size and by changing the nature of the speech material. The first recognition task was called Resource Management (RM). The data for this task consisted of read queries on the status of U.S. naval resources, using a vocabulary of about 1,000 words and a fairly constrained syntax (giving a test-set perplexity of about 60). Recognition tests were first carried out on this task in 1987. Formal evaluations began in 1989 and ended in 1992. New testing data were provided for each of the evaluations, and each year the recognition performance improved as shown in Figure 12.6.

The next ARPA evaluations focused on recognition of spoken newspaper texts. This application allowed much larger-vocabulary tasks to be studied and also ensured that there was easy availability of large quantities of text data for training statistical language models. Initially the texts were all obtained from the *Wall Street*

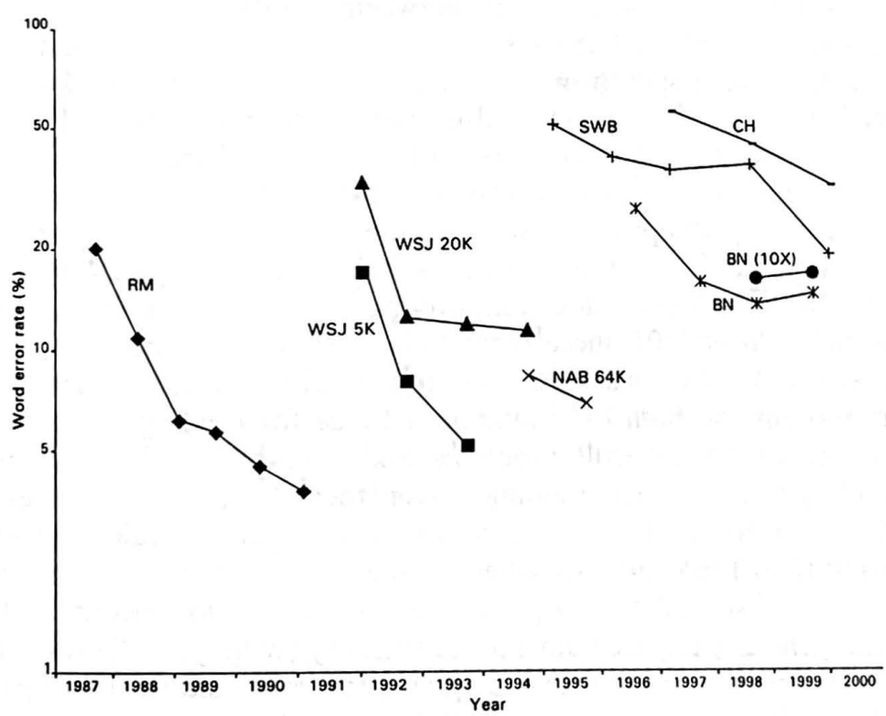


Figure 12.6 Graph showing recognition performance (in terms of word error rate, plotted on a logarithmic scale) for various transcription tasks used in ARPA evaluations. Each data point has been taken from published results for the relevant evaluation, with the aim of showing the performance of the system that was the best on that particular test. Note that, even within any one task, different evaluation data have been used each year, and in some cases there were differences in the task details. Thus, while the lines in the graph give an indication of general trends in performance, strict comparisons of performance in successive years are more difficult.

Journal (WSJ), and included both a 5,000-word recognition test and a 20,000-word test. More recently, the test set was expanded to include a range of North American business (NAB) news and restrictions on training for the acoustic and language models were removed. The recognition vocabulary became essentially unlimited and performance was generally evaluated using systems designed for recognizing about 64,000 words. WSJ and NAB evaluations were conducted in the period between 1992 and 1995. Over the years the complexity and variety of the tests increased, but recognition performance still improved each year (see Figure 12.6).

Later ARPA evaluations moved away from read speech, and concentrated on transcribing 'found' speech from broadcast news (BN) shows on radio and television. Such data provided a lot of variation in speech quality, microphones and background noise, as well as non-native speech and periods containing only non-speech data such as sound effects and music. For the first BN evaluation in 1996, performance dropped considerably from that which had been obtained in the previous evaluations using read speech. However, researchers soon found ways of coping with the greater variety of material, and by the final BN evaluations in 1998 and 1999 typical error rates had roughly halved from those obtained in 1996.

ARPA have also sponsored a research programme on the recognition of conversational speech, including the collection of two different corpora. Data for one corpus, known as Switchboard (SWB), first became available in 1992. The data comprised telephone conversations between two strangers about some topic that had been specified in advance. A few years later, in 1995, collection of the Callhome (CH) corpus was initiated to provide more natural conversational speech by recording telephone conversations between family members. The CH set includes a number of different languages. Conversational speech is rather different in nature to either read speech or television and radio broadcasts. Conversations tend to include many disfluencies, false starts and so on, as well as a lot of phonological reduction and special use of prosody. There is also turn-taking behaviour and reliance on shared knowledge to assist in the communication, especially when the participants know each other very well (as in the CH data). As a result, errors are much higher than for the other tasks (see Figure 12.6), but again substantial progress has been made in improving performance on the conversational recognition tasks. As of 2001, these evaluations are still continuing.

The ARPA initiative has been very influential in the development of large-vocabulary recognizers, both by providing a focus for refining the techniques to cope with increasingly difficult problems and also by making available huge quantities of speech data for training. Over the years, the result has been a progressive improvement in performance on increasingly difficult tasks, as can be seen in Figure 12.6. There are now several research systems which give impressive performance on substantial transcription tasks. However, to perform at their best these systems generally require both a lot of memory (with typically several million parameters in both the acoustic and language models), and a lot of processing power (often operating at a few hundred times real time). In the last two BN evaluations, the effect of processing demands was tested by introducing a category for systems that operated at no slower than 10 times real time (shown as BN(10×) in Figure 12.6). Enforcing this constraint led to around 15% more errors than if there were no processing restrictions. We will consider performance in relation to the requirements of different applications in Chapter 15.

IPR2023-00037
Apple EX1016 Page 97

12.10 SPEECH UNDERSTANDING

In order to understand spoken language, it is necessary not only to recognize the words, but also to determine the linguistic structure, including both syntactic and semantic aspects. Given a word sequence, the field of computational linguistics provides techniques for deriving a representation of the linguistic structure. Traditionally these techniques involve a complex linguistic analysis based on qualitative models that are usually stated in the form of a grammar that is specified manually by human experts, although in recent years data-driven statistical methods have also been used. Even so, any detailed linguistic analysis will involve modelling long-range effects that may operate over the entire span of an utterance. It is very difficult to incorporate such a model into a speech recognition search.

Due to the difficulties involved in attempting to incorporate detailed linguistic analysis into the recognition search, speech understanding systems generally treat these tasks as separate components: an ASR system of the type described earlier in this chapter, and a second component that uses artificial-intelligence techniques to perform linguistic analysis in order to 'understand' the recognizer output. To reduce the problems caused by errors made at the recognition stage, the output of the recognizer should include alternatives, which can be provided in the form of a word lattice or an N-best list. The final recognized output is typically taken to be the highest-scoring alternative that is also allowed by the linguistic model.

The process of determining the linguistic structure of a sentence is known as **parsing**. Syntactic structure is usually expressed in terms of a formal grammar, and there are a variety of grammar formalisms and associated methods for **syntactic parsing**. However, traditional parsers aim to recover complete, exact parses. This goal will often not be achievable for spoken language, which tends to contain grammatical errors as well as hesitations, false starts and so on. The problem is made even worse when dealing with the output of a speech recognizer, which may misrecognize short function words even when overall recognition performance is very good. It can therefore be useful to adopt **partial parsing** techniques, whereby only segments of a complete word sequence are parsed (for example, noun phrases might be identified). Other useful tools include **part-of-speech taggers**, which aim to disambiguate parts of speech (e.g. the word "green" acts as an adjective in the phrase "green coat" but as a noun in "village green"), but without performing a parsing operation. Some taggers are rule-based, but there are also some very successful taggers that are based on HMMs, with the HMM states representing tags (or sequences of tags). Transition probabilities are probabilities of tag(s) given previous tag(s) and emission probabilities are probabilities of words given tags. Partial parsing and part-of-speech tagging enable useful linguistic information to be extracted from spoken input without requiring a comprehensive linguistic analysis.

General **semantic analysis** for the derivation of meaning representations is a complex aspect of computational linguistics. However, many speech-understanding systems have used a simpler approach that is more application dependent. A popular technique uses templates, or 'frames', where each frame is associated with an action to be taken by the system. A frame has 'slots' for different items of relevant information. In a flight enquiry system for example, one frame could represent a request for flight information. This frame could include slots for the type of information (e.g. flights, fares) and for any constraints (e.g. date, price).

The task is to find the frame that gives the best match to the input, and to obtain values for the slots in that frame. Once a frame has been recognized and its slots have been filled, the system's response can be determined.

Many applications of speech understanding involve the user asking questions in order to access information from a computer system. Often a query may be insufficiently specified or ambiguous. However, by entering into an interactive dialogue with the user, the system should be able to obtain the additional information or resolve any ambiguities. A **dialogue manager** can greatly assist in the usability of these **spoken dialogue systems**, by keeping track of the interaction, guiding the user and asking questions where necessary.

12.10.1 Measuring and evaluating speech understanding performance

When evaluating speech understanding systems, word and sentence recognition errors are obviously of interest. However, some measure of a system's ability to recognize meaning is also required. This measure is usually the percentage of utterances for which the complete system gives an acceptable output, as judged by a trained human operator. For a spoken dialogue system that is intended for a particular application, it may be most informative to evaluate performance using task-based measures such as the percentage of tasks successfully completed and time taken to accomplish the tasks, together with measures of user satisfaction.

In parallel with its speech transcription evaluations, ARPA has also been conducting a series of evaluations of spoken language understanding systems. In the late 1980s, work began on an Air Travel Information System (ATIS) task, and formal evaluations were carried out in the period between 1990 and 1994. The ATIS task involved recognizing and responding to spontaneous queries about airline reservations (e.g. "List the flights from Boston to Dallas." or "Is lunch served on that flight?"), using a vocabulary of about 2,500 words. In 1994, the best speech recognition performance was a word error rate of 2.3% (Moore *et al.*, 1995), which improved to 1.9% when the output of the speech recognizer was subjected to post-processing by a natural-language system that was tuned to the application domain. Although this task involves spontaneous speech, the constraints of the task domain enable high recognition accuracy to be achieved. The best understanding performance, taken over all the answerable spoken queries, was an error rate of 8.6% (Pallett *et al.*, 1995). Although there were more understanding errors than word recognition errors, this level of performance is probably adequate for many database query applications. However, achieving such performance requires the system to be tuned to its particular application domain, with a lot of 'knowledge' that is specific to the one domain (flights in the case of the ATIS task).

More recent ARPA evaluations have focused on other types of tasks, which involve processing large quantities of speech material in a way that requires some understanding capability. Past and ongoing tasks include retrieval of spoken documents, detection and tracking of topics in broadcasts, and extraction of the content (meaning) from transcripts of spoken news material. Automatic speech understanding is a pre-requisite for all of the most advanced applications of spoken language processing, of which one of the hardest is automatic speech translation (see Chapters 15 and 16 for further discussion).

CHAPTER 12 SUMMARY

- Some large-vocabulary recognition tasks may require accurate transcription of the words that have been said, while others will need understanding of the semantic content (but not necessarily accurate recognition of every word).
- For speech transcription the task is to find the most likely sequence of words, where the probability for any one sequence is given by the product of acoustic-model and language-model probabilities.
- The principles of continuous speech recognition using HMMs can be applied to large vocabularies, but with special techniques to deal with the large number of different words that need to be recognized.
- It is not practical or useful to train a separate model for every word, and instead sub-word models are used. Typically phone-size units are chosen, with the pronunciation of each word being provided by a dictionary.
- Triphone models represent each phone in the context of its left and right neighbours. The large number of possible triphones is such that many will not occur in any given set of training data. Probabilities for these triphones can be estimated by 'backing off' or interpolating with biphones (dependent on only the left or the right context) or even context-independent monophones.
- Another option, which allows greater context specificity to be achieved, is to group ('cluster') similar triphones together and share ('tie') their parameters. A phonetic decision tree can be used to find the best way to cluster the triphones based on questions about phonetic context. The idea is to optimize the fit to the data while also having sufficient data available to train each tied state.
- An embedded training procedure is used, typically starting by estimating parameters for very general monophone models for which a lot of data are available. These models are used to initialize triphone models. The triphones are trained and similar states are then tied together. Multiple-component mixture distributions are introduced at the final stage.
- The purpose of the language model is to incorporate language constraints, expressed as probabilities for different word sequences. The perplexity, or average branching factor, provides a measure of how good the language model is at predicting the next word given the words that have been seen so far.
- N-grams model the probability of a word depending on just the immediately preceding $N-1$ words, where typically $N=2$ ('bigrams') or $N=3$ ('trigrams').
- The large number of different possible words is such that data sparsity is a massive problem for language modelling, and special techniques are needed to estimate probabilities for N-grams that do not occur in the training data.
- Probabilities for N-grams that occur in the training text can be estimated from frequency counts, but some probability must be 'freed' and made available for those N-grams that do not occur. Probabilities for these unseen N-grams can then be estimated by backing off or interpolating with more general models.
- The principles of HMM recognition extend to large vocabularies, with a multiple-level structure in which phones are represented as networks of states, words as networks of phones, and sentences as networks of words. In practice the decoding task is not straightforward due to the very large size of the search space, especially if cross-word triphones are used. Special treatment is also required for language models whose probabilities depend on more than the

immediately preceding word (i.e. for models more complex than bigrams). The one-pass Viterbi search can be extended to operate with cross-word triphones and with trigram language models, but the search space becomes very large and is usually organized as a tree. Efficient pruning is essential.

- An alternative search strategy uses multiple passes. The first pass identifies a restricted set of possibilities, which are typically organized as an *N*-best list, a word lattice or a word graph. Later passes select between these possibilities. Another option is to use a depth-first search.
- Automatic speech understanding needs further processing of the speech recognizer output to analyse the meaning, which may involve syntactic and semantic analysis. To reduce the impact of recognition errors, it is usual to start with an *N*-best list or word lattice. Partial parsing techniques can be used for syntactic analysis to deal with the fact that the spoken input may be impossible to parse completely because parts do not fit the grammar, due to grammatical errors, hesitations and so on.
- Meaning is often represented using templates, which will be specific to the application and have 'slots' that are filled by means of a linguistic analysis.
- In spoken dialogue systems, a dialogue manager is used to control the interaction with the user and ensure that all necessary information is obtained.
- ARPA has been influential in promoting progress in large vocabulary recognition and understanding, by sponsoring the collection of large databases and running series of competitive evaluations. Error rates of less than 10% have been achieved for transcribing unlimited-vocabulary read speech and for understanding spoken dialogue queries. Recognition of more casually spoken spontaneous speech is still problematic.

CHAPTER 12 EXERCISES

E12.1 Explain the different requirements and problems in speech transcription and speech understanding.

E12.2 What are the special issues for the design of the acoustic model, the language model and the search component when a recognizer needs to cope with a large vocabulary size?

E12.3 Give examples of how acoustic/phonetic knowledge can be used when choosing an acoustic model set for a large-vocabulary recognition system.

E12.4 When estimating trigram language-model probabilities based on counts in some training text, explain how probabilities can be estimated for trigrams that do not occur in the training data. How does this operation affect the probabilities that need to be assigned to trigrams that do occur?

E12.5 What are the similarities and differences between the concept of 'backing off' in language modelling and in acoustic modelling?

E12.6 Explain the relative merits of performing large-vocabulary recognition using a one-pass Viterbi search versus using a multiple-pass search strategy.

E12.7 What measures are required to evaluate the performance of a speech understanding system?

With the growing impact of information technology on daily life, speech is becoming increasingly important for providing a natural means of communication between humans and machines. This extensively reworked and updated new edition of **SPEECH SYNTHESIS AND RECOGNITION** is an easy-to-read introduction to current speech technology.

Aimed at advanced undergraduates and graduates in electronic engineering, computer science and information technology, the book is also relevant to professional engineers who need to understand enough about speech technology to be able to apply it successfully and to work effectively with speech experts. No advanced mathematical ability is required and no specialist knowledge prior of phonetics or of the properties of speech signals is assumed.

SPEECH SYNTHESIS AND RECOGNITION:

- **explains the complexity of speech communication**
- **describes mechanisms and models of human speech production and perception**
- **covers concatenative synthesis techniques and formant synthesis by rule, as well as the processing required for synthesis from text**
- **introduces methods for automatic speech recognition by whole-word template matching and by statistical pattern matching using hidden Markov models**
- **describes practical techniques that contribute to the successful implementation of speech recognition systems, including those for recognizing very large vocabularies**
- **includes chapters covering the related technologies of digital speech coding and automatic recognition of speaker characteristics**
- **discusses applications and performance of current speech technology**

Throughout the book the emphasis is on explaining underlying principles with sufficient but not unnecessary detail, so as to provide the reader with a thorough grounding in the problems and techniques involved in speech synthesis and recognition. This book is therefore ideal as an introduction before tackling more advanced texts.

Wendy Holmes is a research scientist with 20/20 Speech Ltd in Malvern, England. The late **John Holmes** was an internationally regarded expert in the field of speech research, who spent several years as head of the UK government's Joint Speech Research Unit.

Electrical engineering
Design: www.egelnickandwebb.com

ISBN 0-7484-0857-6
9 780748 408573

11 New Fetter Lane
London EC4P 4EE
29 West 35th Street
New York NY10001
www.tandf.co.uk
Printed in Great Britain

TAYLOR & FRANCIS