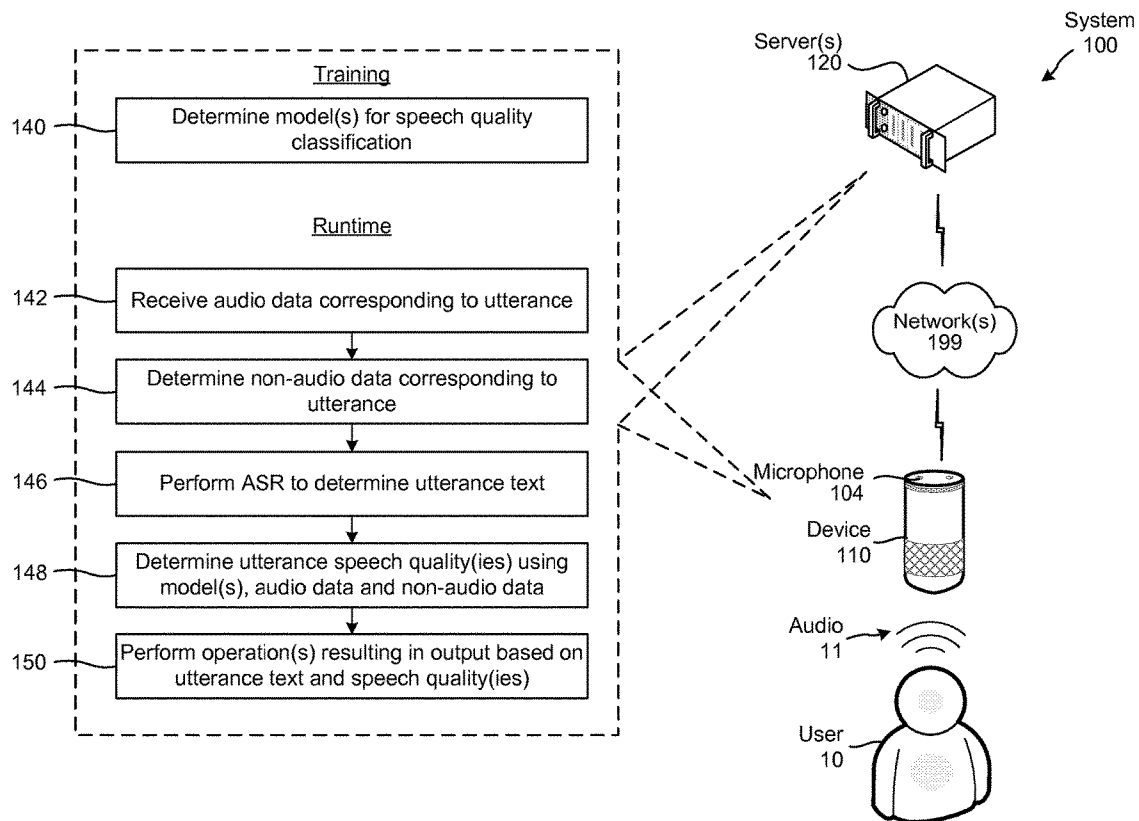




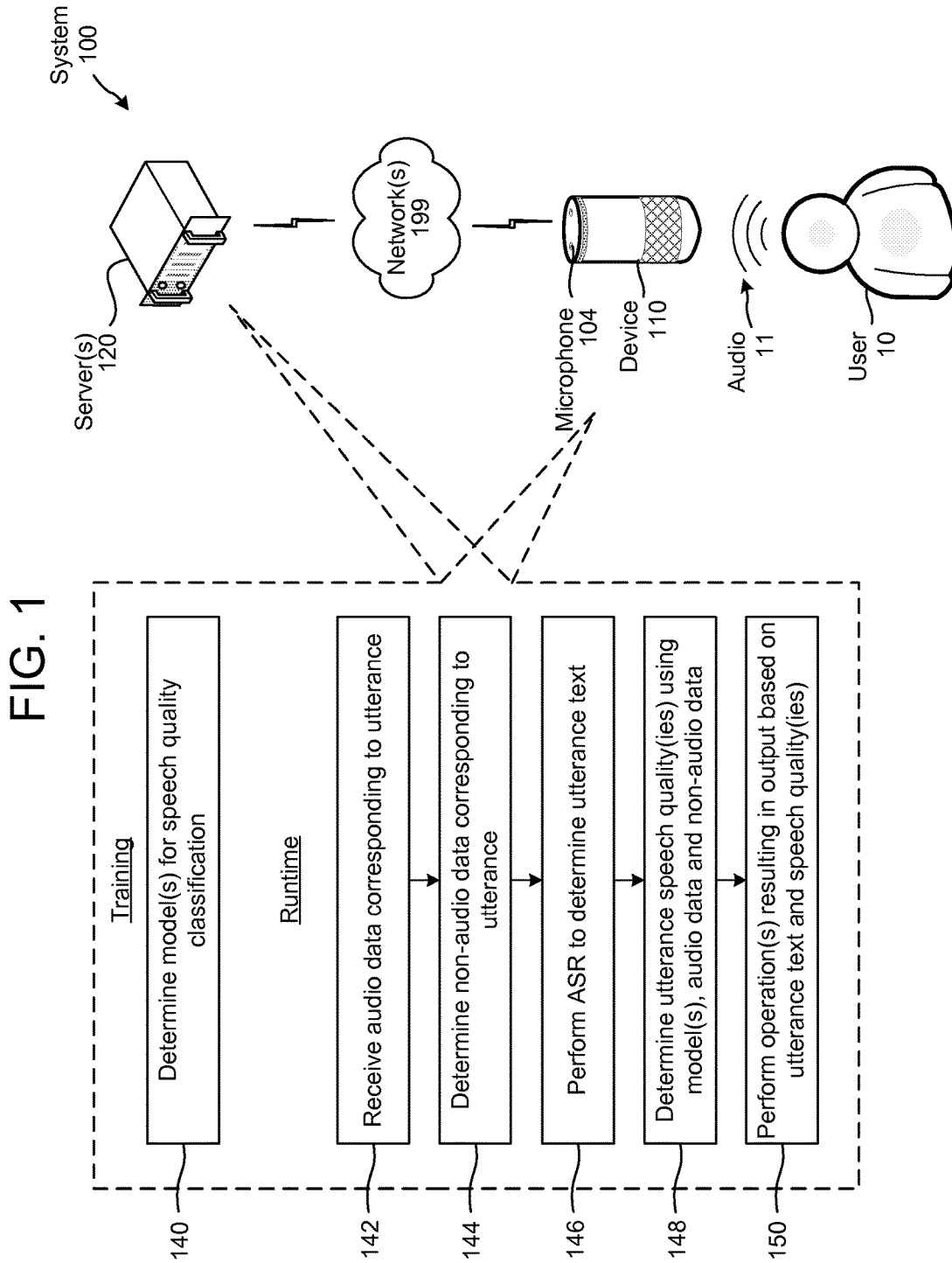
US 20160379638A1

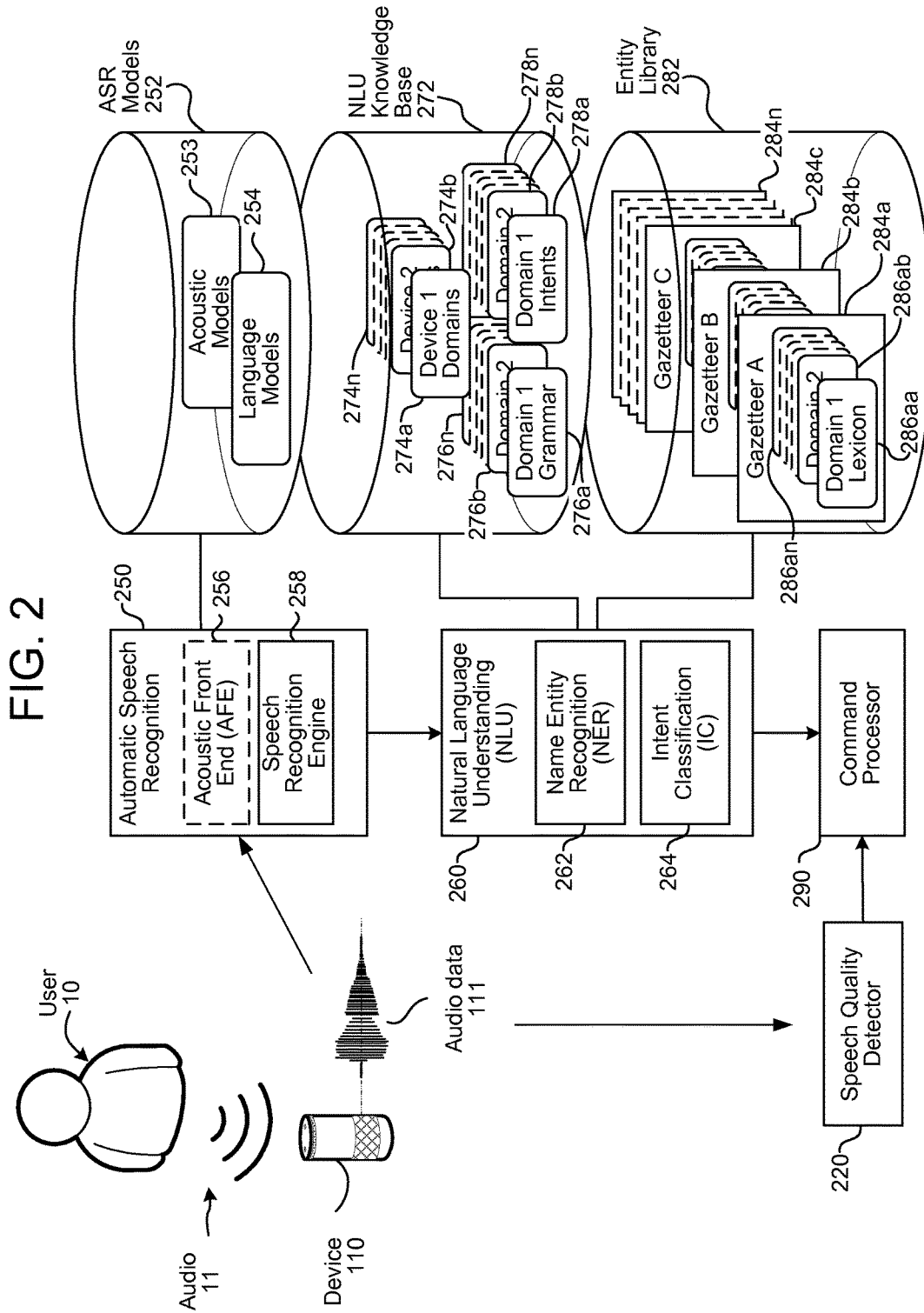
(19) **United States**(12) **Patent Application Publication**
Basye et al.(10) **Pub. No.: US 2016/0379638 A1**(43) **Pub. Date: Dec. 29, 2016**(54) **INPUT SPEECH QUALITY MATCHING**(71) Applicant: **Amazon Technologies, Inc.**, Seattle,
WA (US)(72) Inventors: **Kenneth John Basye**, Sutton, MA
(US); **Arthur Richard Toth**,
Somerville, MA (US); **William Folwell**
Barton, Harvard, MA (US)(21) Appl. No.: **14/752,128**(22) Filed: **Jun. 26, 2015****Publication Classification**(51) **Int. Cl.**
G10L 15/26 (2006.01)
G10L 15/18 (2006.01)
G10L 13/02 (2006.01)(52) **U.S. Cl.**CPC **G10L 15/26** (2013.01); **G10L 13/02**
(2013.01); **G10L 15/18** (2013.01)(57) **ABSTRACT**

A system matches text-to-speech (TTS) or other output to a quality of an input spoken utterance. The system uses trained models to detect a speech quality and generates an indicator of the speech quality. The speech quality may be determined from audio or non-audio data. The indicator is sent to downstream components of the system such as a command processor or TTS system. The output of the system is then determined using the indicator of speech quality, thus customizing an output of the system to the manner in which the utterance was spoken.



Amazon v. SoundClear
US Patent 11,244,675
Amazon EX-1208





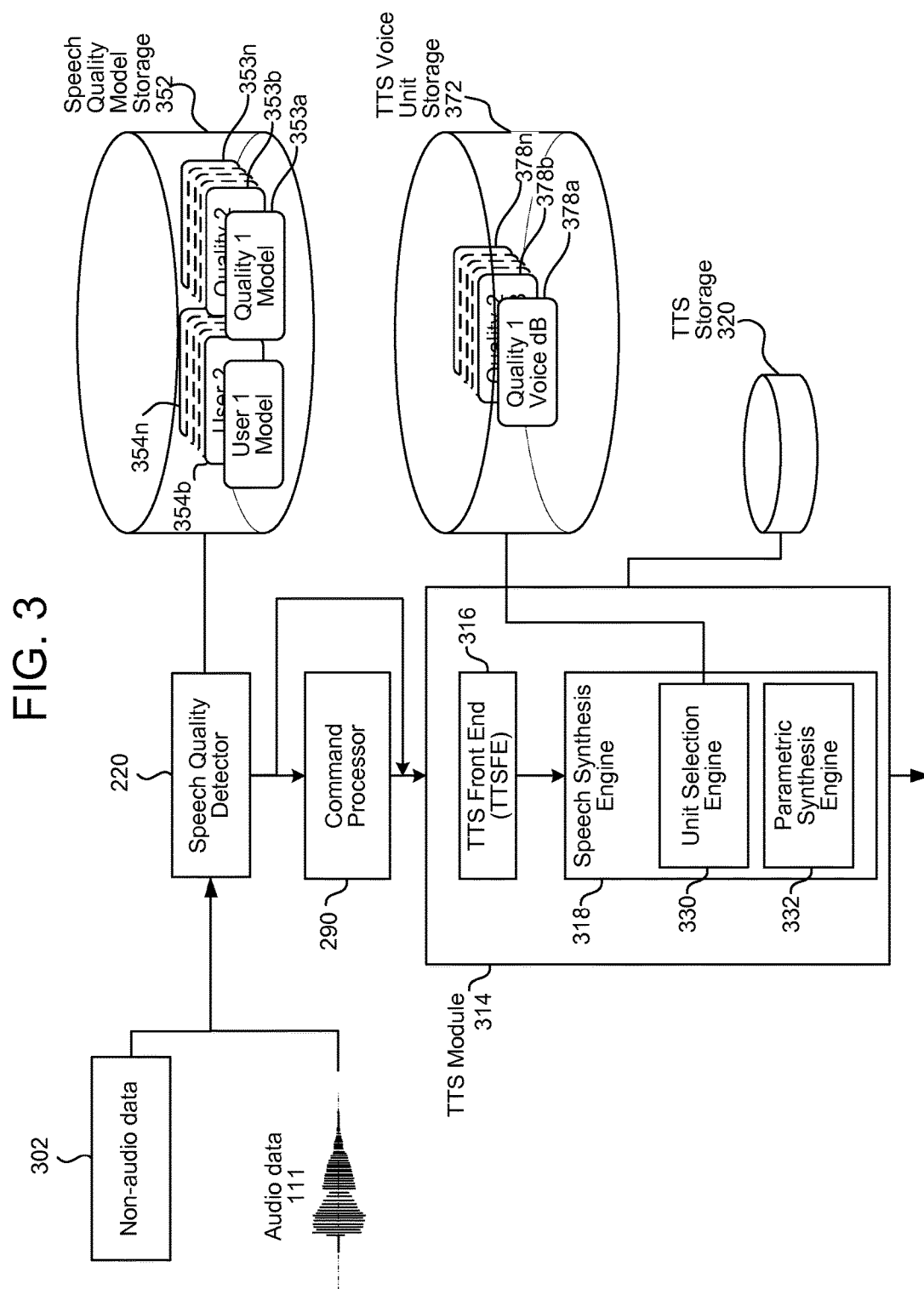


FIG. 4

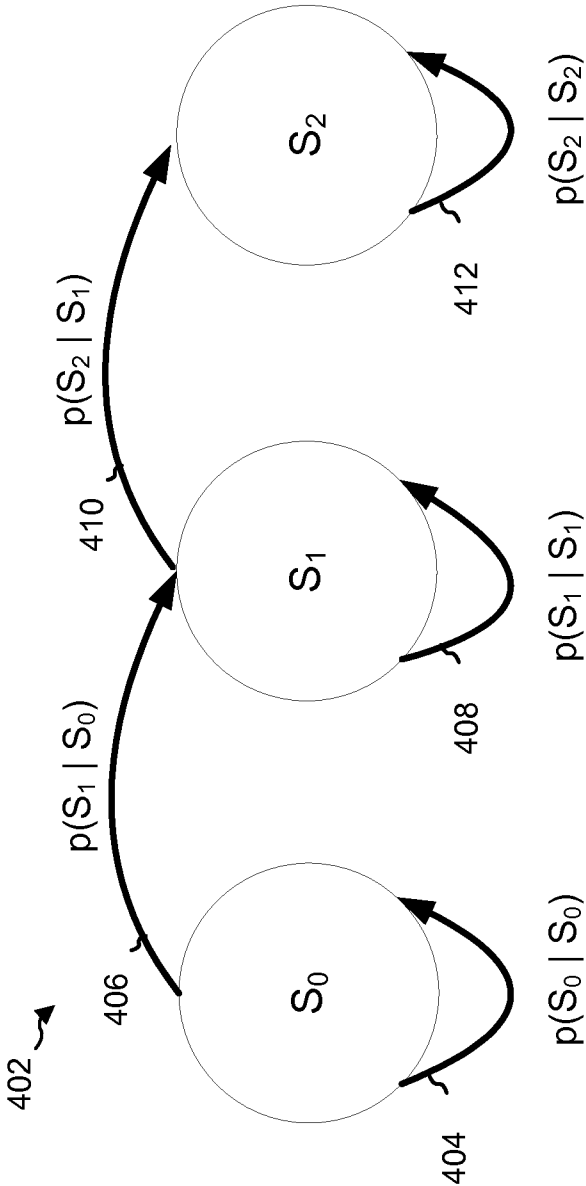


FIG. 5A

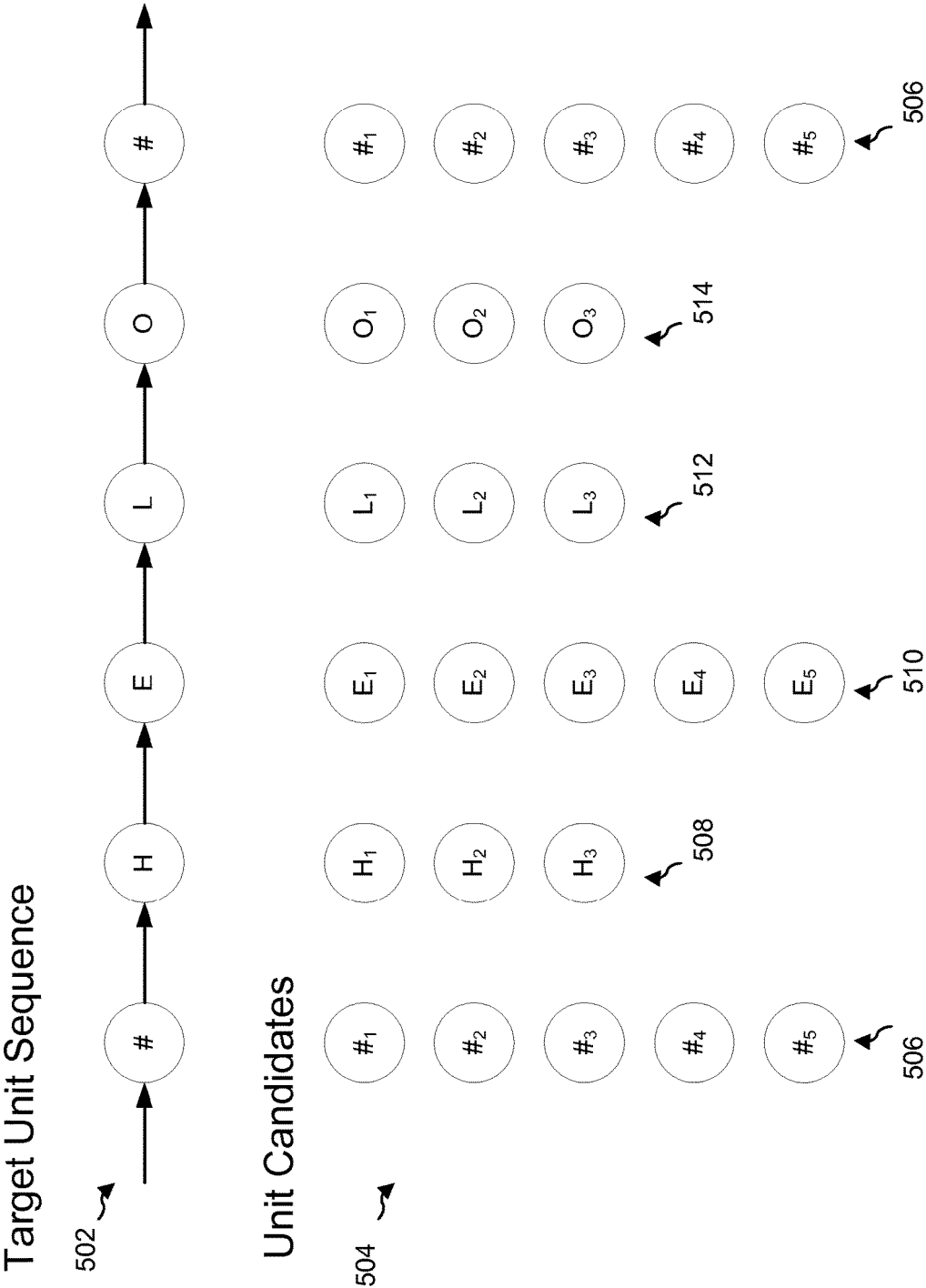


FIG. 5B

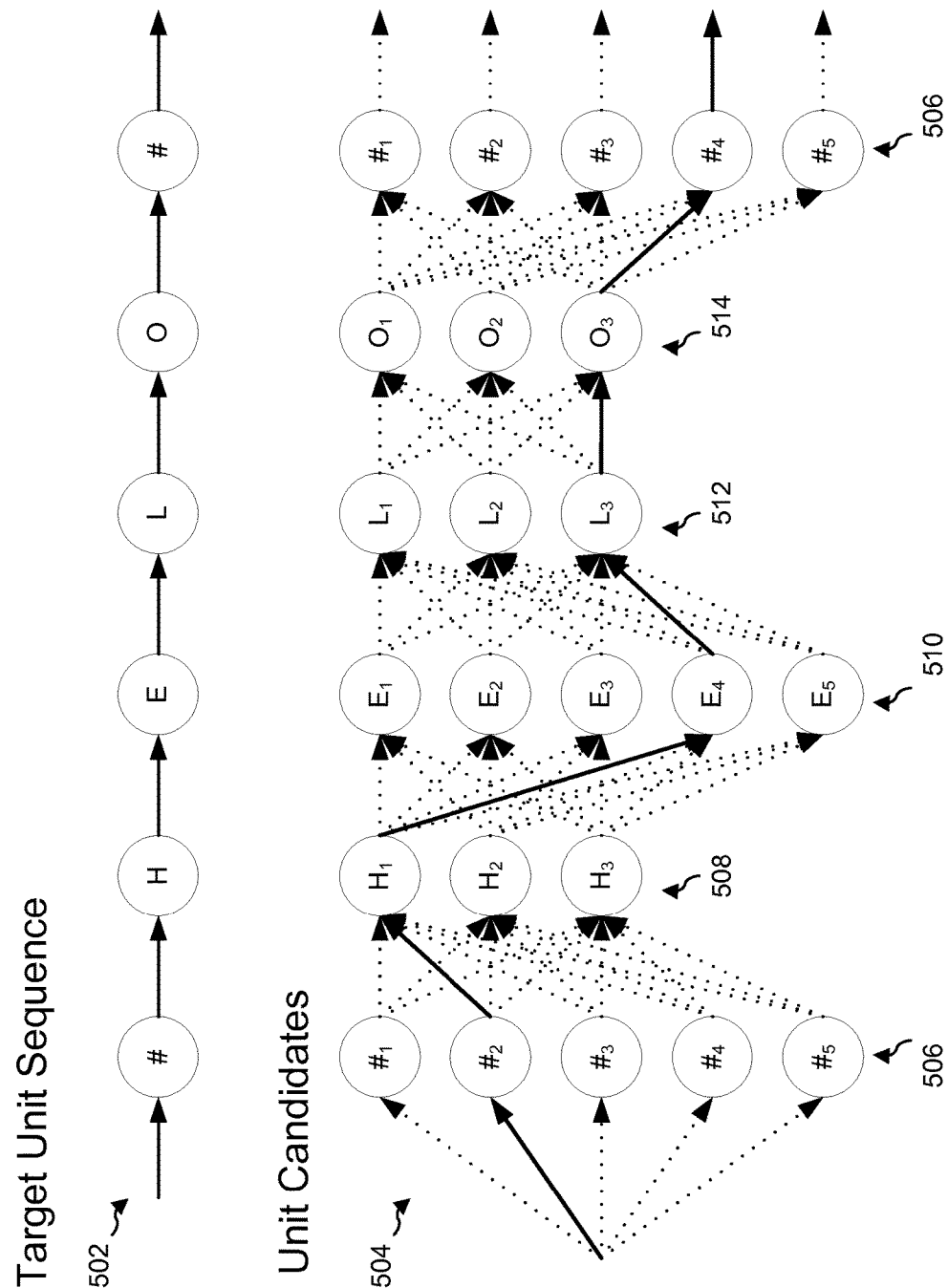


FIG. 6A

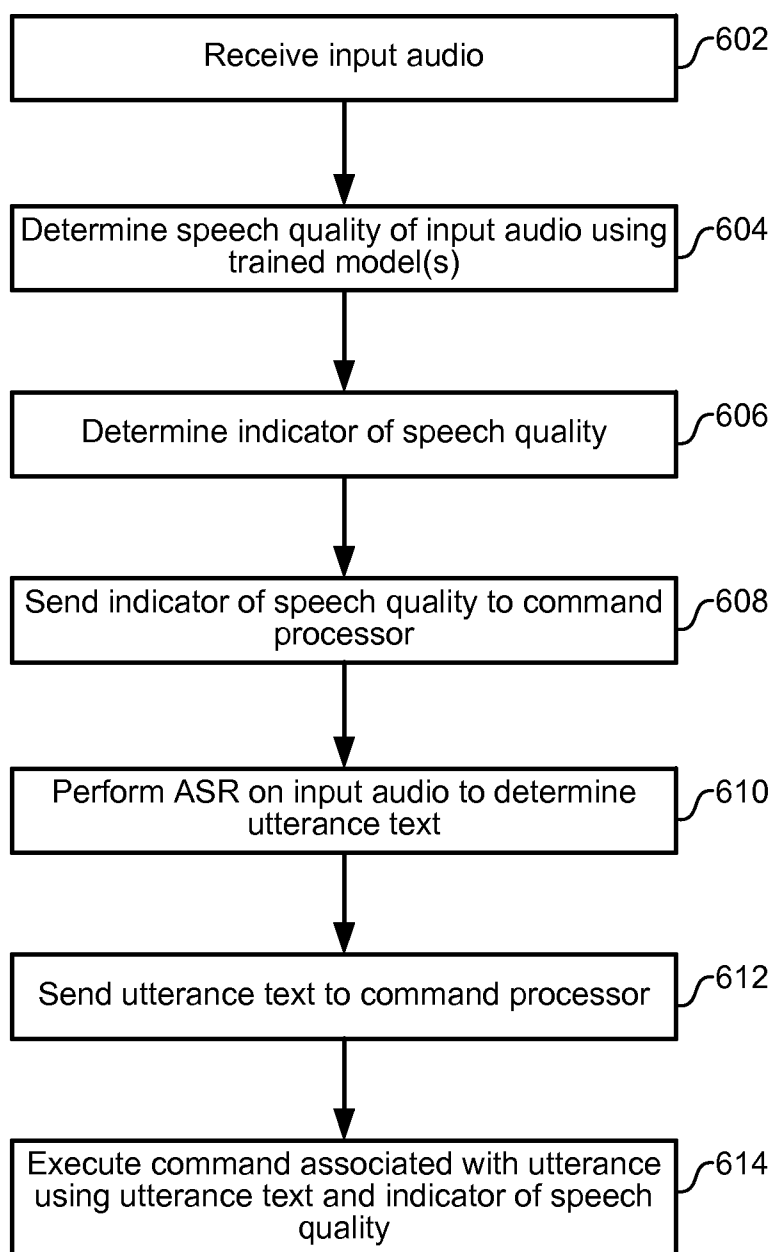


FIG. 6B

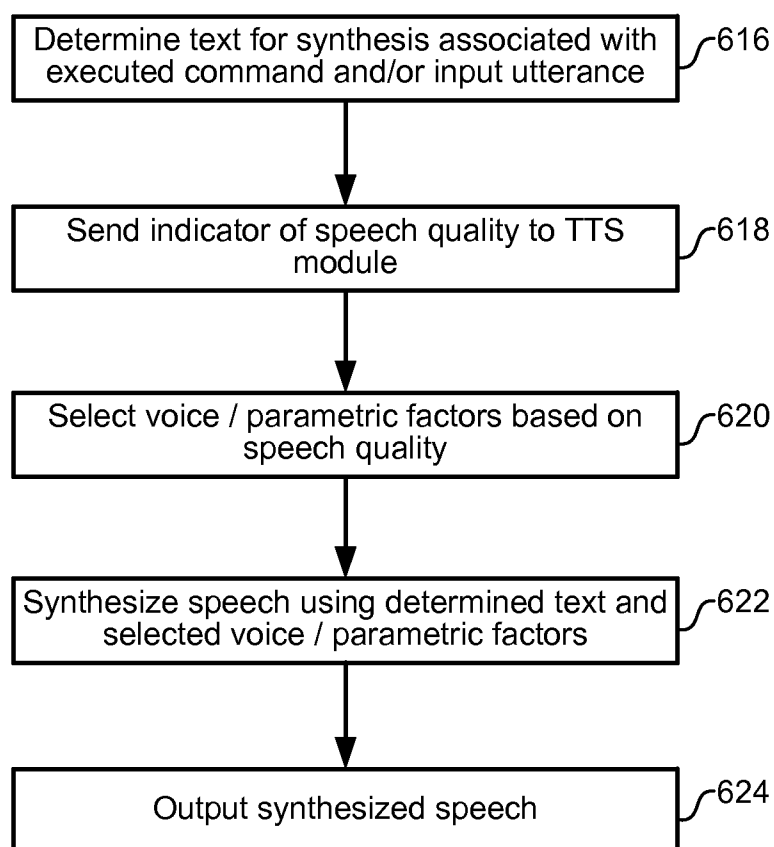


FIG. 7

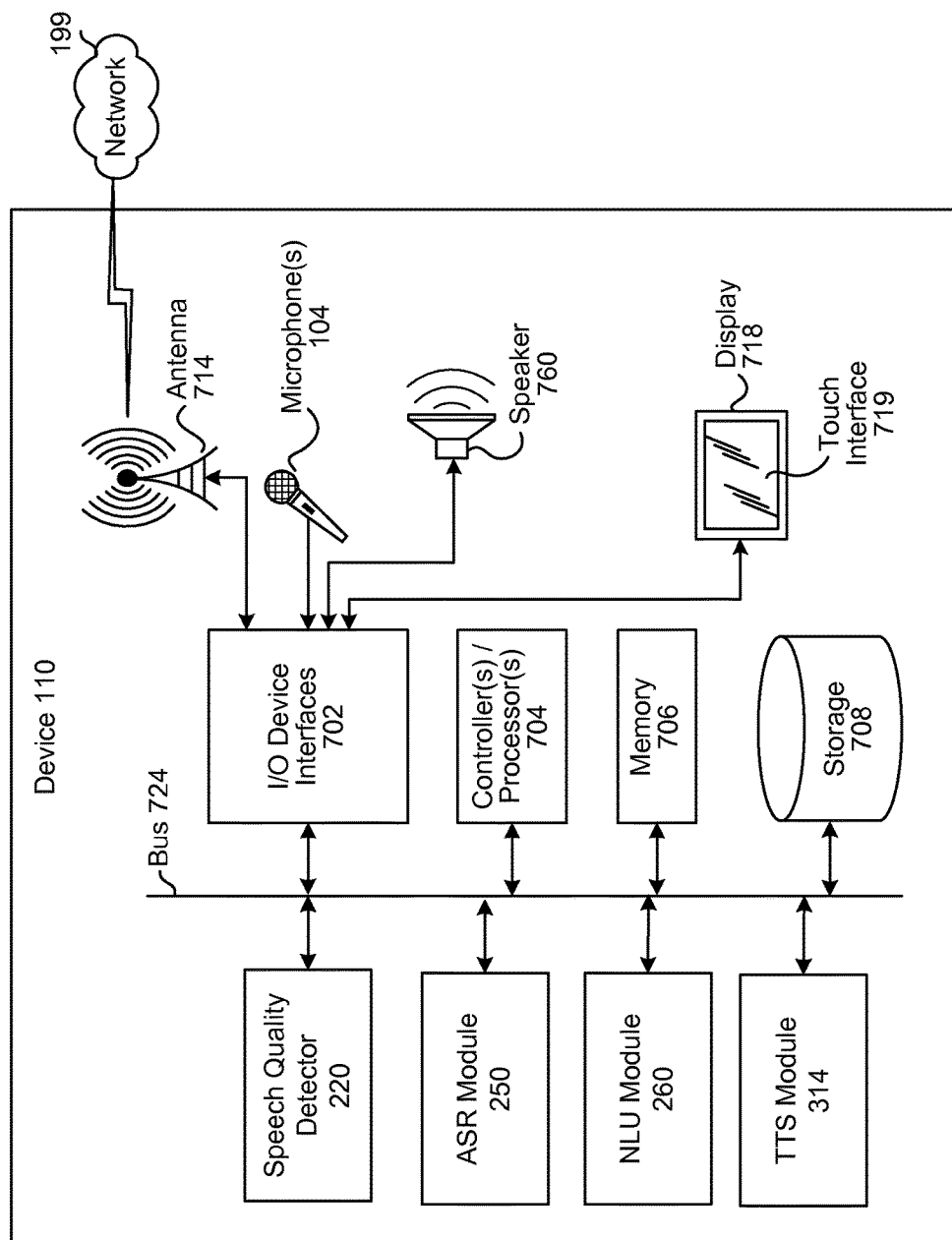
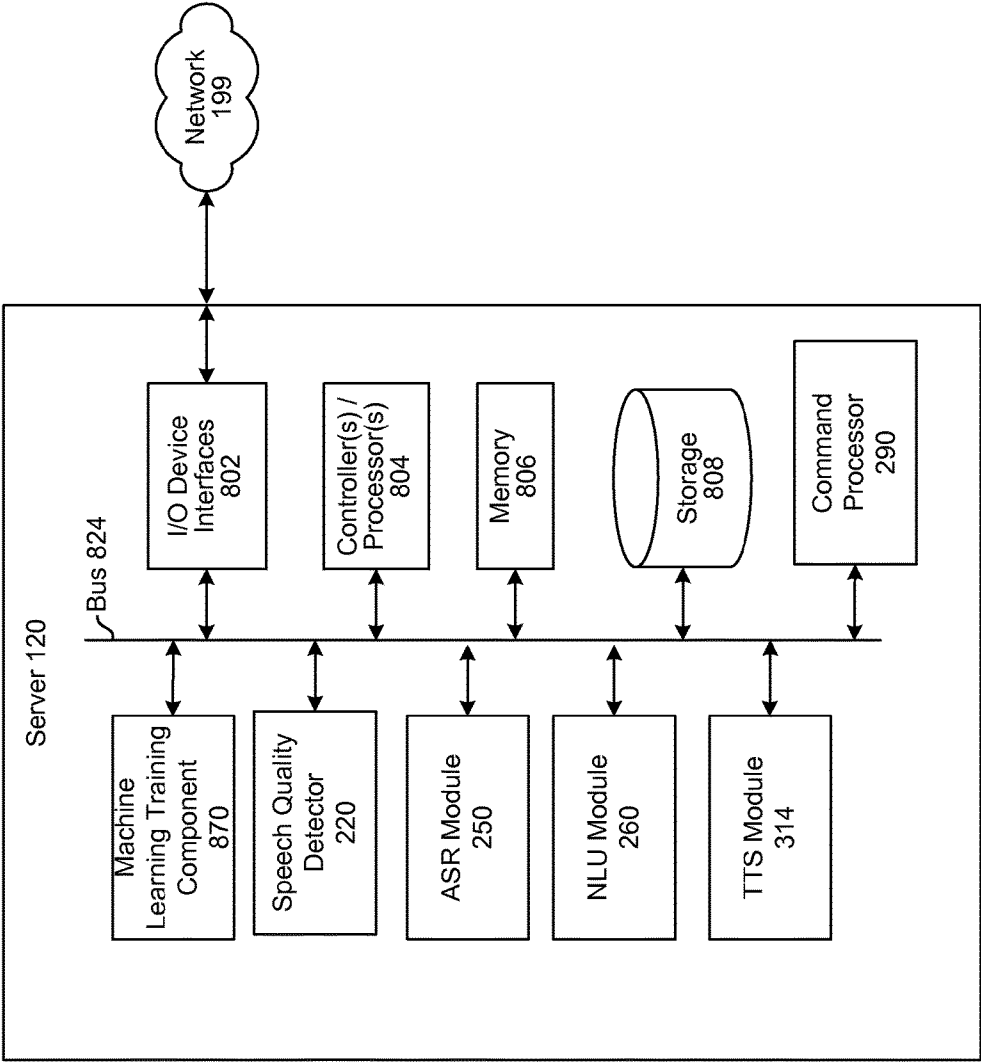
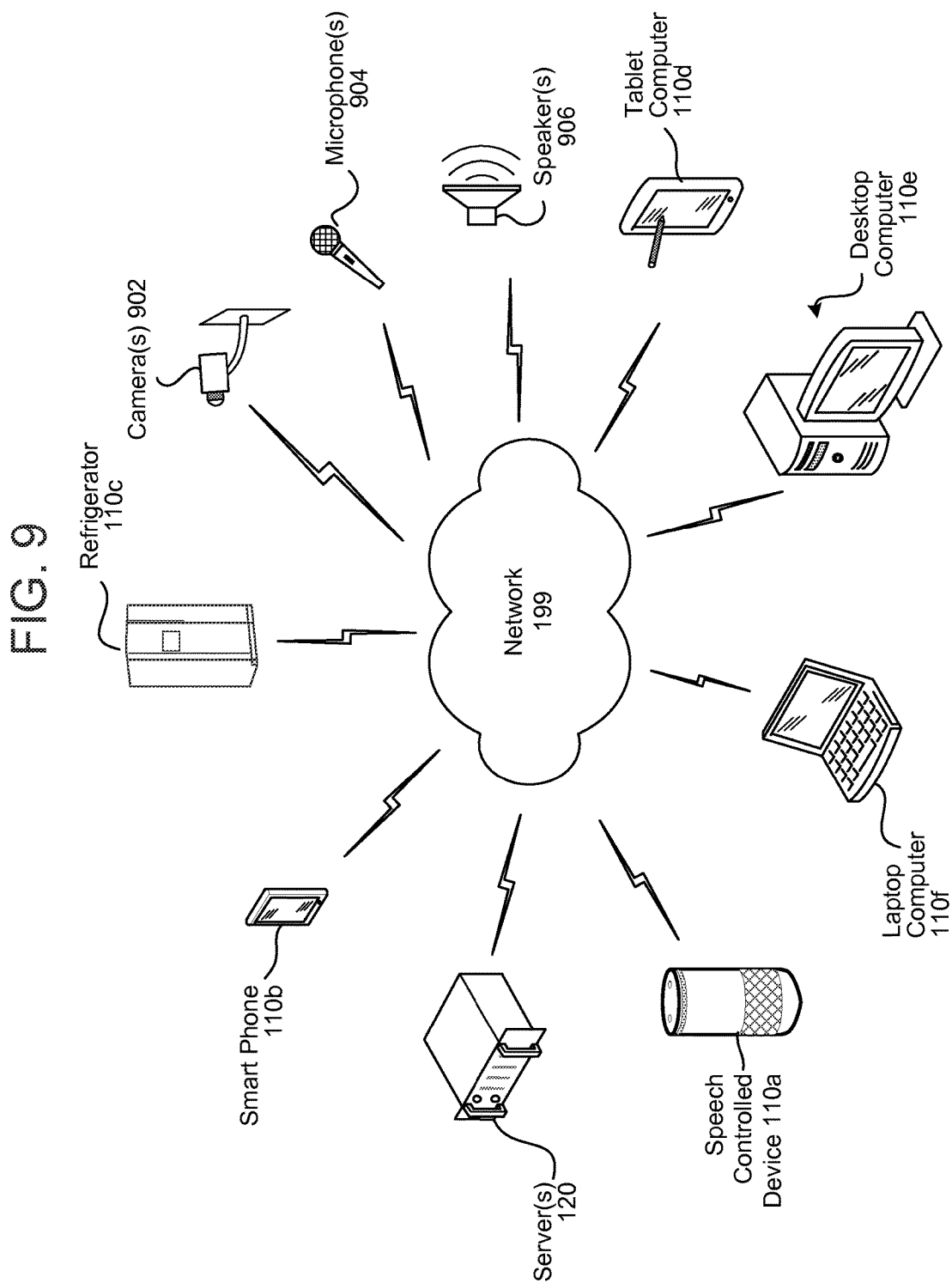


FIG. 8





INPUT SPEECH QUALITY MATCHING

BACKGROUND

[0001] Speech recognition systems have progressed to the point where humans can interact with computing devices entirely relying on speech. Such systems employ techniques to identify the words spoken by a human user based on the various qualities of a received audio input. Speech recognition combined with natural language understanding processing techniques enable speech-based user control of a computing device to perform tasks based on the user's spoken commands. The combination of speech recognition and natural language understanding processing techniques is commonly referred to as speech processing. Speech processing may also convert a user's speech into text data which may then be provided to various text-based software applications.

[0002] Speech processing may be used by computers, hand-held devices, telephone computer systems, kiosks, and a wide variety of other devices to improve human-computer interactions.

BRIEF DESCRIPTION OF DRAWINGS

[0003] For a more complete understanding of the present disclosure, reference is now made to the following description taken in conjunction with the accompanying drawings.

[0004] FIG. 1 illustrates a system for keyword recognition according to embodiments of the present disclosure.

[0005] FIG. 2 is a conceptual diagram of how a spoken utterance may be processed according to embodiments of the present disclosure.

[0006] FIG. 3 is a conceptual diagram of how speech quality may be determined and used to determine a command output or text-to-speech output of a system.

[0007] FIG. 4 illustrates speech synthesis using a Hidden Markov Model according to one aspect of the present disclosure.

[0008] FIGS. 5A-5B illustrate speech synthesis using unit selection according to one aspect of the present disclosure.

[0009] FIGS. 6A-6B are flow diagrams illustrating matching system output to an input speech quality.

[0010] FIG. 7 is a block diagram conceptually illustrating example components of a device according to embodiments of the present disclosure.

[0011] FIG. 8 is a block diagram conceptually illustrating example components of a server according to embodiments of the present disclosure.

[0012] FIG. 9 illustrates an example of a computer network for use with the system.

DETAILED DESCRIPTION

[0013] Automatic speech recognition (ASR) is a field of computer science, artificial intelligence, and linguistics concerned with transforming audio data associated with speech into text representative of that speech. Similarly, natural language understanding (NLU) is a field of computer science, artificial intelligence, and linguistics concerned with enabling computers to derive meaning from text input containing natural language. ASR and NLU are often used together as part of a speech processing system. Text-to-speech (TTS) is a field of concerning transforming textual data into audio data that is synthesized to resemble human speech.

[0014] An increasing number of devices, including home appliances, are becoming capable of processing spoken commands using ASR processing. Further, an increasing number of devices are capable of providing output to users in the form of synthesized speech using TTS processing. When interactions with a device involve spoken technology it may improve a user experience for human-device interactions to mimic human-human interactions where possible. For example, during a conversation, one human may match a conversational characteristic of another human to which he/she is speaking. When one party to a conversation is animatic and energetic, his conversation partner may increase her energy as a natural part of conversation. Conversely, if one party speaks in a whisper, the other party may respond in a whisper naturally, without ever being asked.

[0015] This speech quality matching does not take place during an exchange with an electronic device. Even if a device is equipped with ASR, TTS, or other speech-based capabilities, a device will not detect a speech quality of input speech and attempt to match that speech quality in an output, whether that output is synthesized speech or some other form of output, such as a command execution. As an example, if a user whispers a command to a device, the device may respond at a default volume setting, which can be a jarring experience for a user. Even if a device is configured to respond at a volume that matches the input speech, it still may be jarring to hear the voice of a happy toned conversationalist coming from a device when the user was speaking in a whisper.

[0016] Offered is a system and method for detecting a speech quality of an utterance using one or more paralinguistic features, for example tone or pitch of voice, whether speech is whining, angry, pleading, etc. The system may then respond to the utterance in a manner that corresponds to the speech quality. For example, when a user whispers a command to a device, the device will not only perform ASR on the command, it will also detect that the command was spoken in whisper. An indicator of the speech being in a whisper will be passed downstream, so that synthesized speech prepared by a TTS engine will also be in a whisper, thus matching the speech quality of the input utterance. Further, the system may interpret generic commands (that is, commands which require some decision making or entity selection on the part of the system) in a manner consistent with the speech quality. For example, a spoken command of "play some music" may be interpreted different by a system if spoken in a scream (which may cause the system to play loud music) than if spoke in a whisper (which may cause the system to play softer music). Other embodiments are also possible.

[0017] An example of the system is described in reference to FIG. 1. As shown in FIG. 1, a system 100 may be configured to respond to an utterance based on a speech quality detected associated with the utterance. As shown, a speech controlled device 110 equipped with one or more microphones 104 is connected over a network 199 to one or more servers 120. The device 110 is configured to detect audio 11 associated with a spoken utterance from user 10. The device may then send audio data associated with the audio 11 to the server 120 for further processing, including analyzing the audio data to classify the utterance and to respond to the utterance, for example by executing a command, determining a synthesized speech output, or the like.

[0018] To perform these operations, during a training phase the system may determine (140) one or more models, such as machine learning models that may be used for speech quality classification, that is to classify the incoming speech as having one or more qualities, for example, whether the speech is whispered. As detailed below, the model(s) may determine speech qualities based on audio data and/or non-audio data and may also be customized based on the user 10 associated with the audio being processed.

[0019] During runtime, the server 120 may receive (142) audio data corresponding to the utterance. The system may also determine (144) non-audio data corresponding to the utterance, for example time data as to when the utterance was received, location data of the utterance, image data associated with the user 10 at the time the utterance was spoken, etc. The system may perform (146) ASR to determine utterance text. The system may then determine (148) one or more utterance speech qualities using the trained model(s), the audio data and the non-audio data. For example, a model configured to determine whether speech was whispered may analyze various audio data feature values to classify the utterance as whispered. The system may then perform (150) one or more operations resulting in output based on the utterance text and the speech quality/ies. For example, if the speech is determined to be whispered and the utterance text corresponds to a request for information (such as the weather) the system may determine the requested information, select a whisper voice (or whisper-like speech parametric factors) to synthesize whispered speech providing the information, send the synthesized whispered speech to the device 110 and output the whispered speech including the information to the user.

[0020] In this manner the system may match the output responding to a spoken command to a speech quality of a spoken command, thus creating a more user friendly interaction with the system.

[0021] Further details of matching output to an input speech quality are explained below, following a discussion of the overall speech processing system of FIG. 2. The system 100 of FIG. 1 may operate using various speech processing components as described in FIG. 2. FIG. 2 is a conceptual diagram of how a spoken utterance is processed. The various components illustrated may be located on a same or different physical devices. Communication between various components illustrated in FIG. 2 may occur directly or across a network 199. An audio capture component, such as a microphone of device 110, captures audio 111 corresponding to a spoken utterance. The device sends audio data 111 corresponding to the utterance, to an ASR module 250. The audio data 111 may be output from an acoustic front end (AFE) 256 located on the device 110 prior to transmission. Or the audio data 111 may be in a different form for processing by a remote AFE 256, such as the AFE 256 located with the ASR module 250.

[0022] An ASR process 250 converts the audio data 111 into text. The ASR transcribes audio data into text data representing the words of the speech contained in the audio data. The text data may then be used by other components for various purposes, such as executing system commands, inputting data, etc. A spoken utterance in the audio data is input to a processor configured to perform ASR which then interprets the utterance based on the similarity between the utterance and pre-established language models 254 stored in

an ASR model knowledge base (ASR Models Storage 252). For example, the ASR process may compare the input audio data with models for sounds (e.g., subword units or phonemes) and sequences of sounds to identify words that match the sequence of sounds spoken in the utterance of the audio data.

[0023] The different ways a spoken utterance may be interpreted (i.e., the different hypotheses) may each be assigned a probability or a confidence score representing the likelihood that a particular set of words matches those spoken in the utterance. The confidence score may be based on a number of factors including, for example, the similarity of the sound in the utterance to models for language sounds (e.g., an acoustic model 253 stored in an ASR Models Storage 252), and the likelihood that a particular word which matches the sounds would be included in the sentence at the specific location (e.g., using a language or grammar model). Thus each potential textual interpretation of the spoken utterance (hypothesis) is associated with a confidence score. Based on the considered factors and the assigned confidence score, the ASR process 250 outputs the most likely text recognized in the audio data. The ASR process may also output multiple hypotheses in the form of a lattice or an N-best list with each hypothesis corresponding to a confidence score or other score (such as probability scores, etc.).

[0024] The device or devices performing the ASR process 250 may include an acoustic front end (AFE) 256 and a speech recognition engine 258. The acoustic front end (AFE) 256 transforms the audio data from the microphone into data for processing by the speech recognition engine. The speech recognition engine 258 compares the speech recognition data with acoustic models 253, language models 254, and other data models and information for recognizing the speech conveyed in the audio data. The AFE may reduce noise in the audio data and divide the digitized audio data into frames representing a time intervals for which the AFE determines a number of values, called features, representing the qualities of the audio data, along with a set of those values, called a feature vector or audio feature vector, representing the features/qualities of the audio data within the frame. Many different features may be determined, as known in the art, and each feature represents some quality of the audio that may be useful for ASR processing. A number of approaches may be used by the AFE to process the audio data, such as mel-frequency cepstral coefficients (MFCCs), perceptual linear predictive (PLP) techniques, neural network feature vector techniques, linear discriminant analysis, semi-tied covariance matrices, or other approaches known to those of skill in the art.

[0025] The speech recognition engine 258 may process the output from the AFE 256 with reference to information stored in speech/model storage (252). Alternatively, post front-end processed data (such as feature vectors) may be received by the device executing ASR processing from another source besides the internal AFE. For example, the device 110 may process audio data into feature vectors (for example using an on-device AFE 256) and transmit that information to a server across a network 199 for ASR processing. Feature vectors may arrive at the server encoded, in which case they may be decoded prior to processing by the processor executing the speech recognition engine 258.

[0026] The speech recognition engine 258 attempts to match received feature vectors to language phonemes and

words as known in the stored acoustic models **253** and language models **254**. The speech recognition engine **258** computes recognition scores for the feature vectors based on acoustic information and language information. The acoustic information is used to calculate an acoustic score representing a likelihood that the intended sound represented by a group of feature vectors matches a language phoneme. The language information is used to adjust the acoustic score by considering what sounds and/or words are used in context with each other, thereby improving the likelihood that the ASR process will output speech results that make sense grammatically.

[0027] The speech recognition engine **258** may use a number of techniques to match feature vectors to phonemes, for example using Hidden Markov Models (HMMs) to determine probabilities that feature vectors may match phonemes. Sounds received may be represented as paths between states of the HMM and multiple paths may represent multiple possible text matches for the same sound.

[0028] Following ASR processing, the ASR results may be sent by the speech recognition engine **258** to other processing components, which may be local to the device performing ASR and/or distributed across the network(s) **199**. For example, ASR results in the form of a single textual representation of the speech, an N-best list including multiple hypotheses and respective scores, lattice, etc. may be sent to a server, such as server **120**, for natural language understanding (NLU) processing, such as conversion of the text into commands for execution, either by the device **110**, by the server **120**, or by another device (such as a server running a search engine, etc.)

[0029] The device performing NLU processing **260** (e.g., server **120**) may include various components, including potentially dedicated processor(s), memory, storage, etc. A device configured for NLU processing may include a named entity recognition (NER) module **252** and intent classification (IC) module **264**, a result ranking and distribution module **266**, and knowledge base **272**. The NLU process may also utilize gazetteer information (**284a-284n**) stored in entity library storage **282**. The gazetteer information may be used for entity resolution, for example matching ASR results with different entities (such as song titles, contact names, etc.) Gazetteers may be linked to users (for example a particular gazetteer may be associated with a specific user's music collection), may be linked to certain domains (such as shopping), or may be organized in a variety of other ways.

[0030] The NLU process takes textual input (such as processed from ASR **250** based on the utterance **11**) and attempts to make a semantic interpretation of the text. That is, the NLU process determines the meaning behind the text based on the individual words and then implements that meaning. NLU processing **260** interprets a text string to derive an intent or a desired action from the user as well as the pertinent pieces of information in the text that allow a device (e.g., device **110**) to complete that action. For example, if a spoken utterance is processed using ASR **250** and outputs the text "call mom" the NLU process may determine that the user intended to activate a telephone in his/her device and to initiate a call with a contact matching the entity "mom."

[0031] The NLU may process several textual inputs related to the same utterance. For example, if the ASR **250** outputs N text segments (as part of an N-best list), the NLU may process all N outputs to obtain NLU results.

[0032] As will be discussed further below, the NLU process may be configured to parse and tag to annotate text as part of NLU processing. For example, for the text "call mom," "call" may be tagged as a command (to execute a phone call) and "mom" may be tagged as a specific entity and target of the command (and the telephone number for the entity corresponding to "mom" stored in a contact list may be included in the annotated result).

[0033] To correctly perform NLU processing of speech input, the NLU process **260** may be configured to determine a "domain" of the utterance so as to determine and narrow down which services offered by the endpoint device (e.g., server **120** or device **110**) may be relevant. For example, an endpoint device may offer services relating to interactions with a telephone service, a contact list service, a calendar/scheduling service, a music player service, etc. Words in a single text query may implicate more than one service, and some services may be functionally linked (e.g., both a telephone service and a calendar service may utilize data from the contact list).

[0034] The name entity recognition module **262** receives a query in the form of ASR results and attempts to identify relevant grammars and lexical information that may be used to construe meaning. To do so, a name entity recognition module **262** may begin by identifying potential domains that may relate to the received query. The NLU knowledge base **272** includes a databases of devices (**274a-274n**) identifying domains associated with specific devices. For example, the device **110** may be associated with domains for music, telephony, calendaring, contact lists, and device-specific communications, but not video. In addition, the entity library may include database entries about specific services on a specific device, either indexed by Device ID, User ID, or Household ID, or some other indicator.

[0035] A domain may represent a discrete set of activities having a common theme, such as "shopping", "music", "calendaring", etc. As such, each domain may be associated with a particular language model and/or grammar database (**276a-276n**), a particular set of intents/actions (**278a-278n**), and a particular personalized lexicon (**286**). Each gazetteer (**284a-284n**) may include domain-indexed lexical information associated with a particular user and/or device. For example, the Gazetteer A (**284a**) includes domain-index lexical information **286aa** to **286an**. A user's music-domain lexical information might include album titles, artist names, and song names, for example, whereas a user's contact-list lexical information might include the names of contacts. Since every user's music collection and contact list is presumably different, this personalized information improves entity resolution.

[0036] A query may be processed applying the rules, models, and information applicable to each identified domain. For example, if a query potentially implicates both communications and music, the query will be NLU processed using the grammar models and lexical information for communications, and will be processed using the grammar models and lexical information for music. The responses based on the query produced by each set of models is scored (discussed further below), with the overall highest ranked result from all applied domains is ordinarily selected to be the correct result.

[0037] An intent classification (IC) module **264** parses the query to determine an intent or intents for each identified domain, where the intent corresponds to the action to be

performed that is responsive to the query. Each domain is associated with a database (278a-278n) of words linked to intents. For example, a music intent database may link words and phrases such as “quiet,” “volume off,” and “mute” to a “mute” intent. The IC module 264 identifies potential intents for each identified domain by comparing words in the query to the words and phrases in the intents database 278.

[0038] In order to generate a particular interpreted response, the NER 262 applies the grammar models and lexical information associated with the respective domain. Each grammar model 276 includes the names of entities (i.e., nouns) commonly found in speech about the particular domain (i.e., generic terms), whereas the lexical information 286 from the gazetteer 284 is personalized to the user(s) and/or the device. For instance, a grammar model associated with the shopping domain may include a database of words commonly used when people discuss shopping.

[0039] The intents identified by the IC module 264 are linked to domain-specific grammar frameworks (included in 276) with “slots” or “fields” to be filled. For example, if “play music” is an identified intent, a grammar (276) framework or frameworks may correspond to sentence structures such as “Play {Artist Name},” “Play {Album Name},” “Play {Song name},” “Play {Song name} by {Artist Name},” etc. However, to make recognition more flexible, these frameworks would ordinarily not be structured as sentences, but rather based on associating slots with grammatical tags.

[0040] For example, the NER module 260 may parse the query to identify words as subject, object, verb, preposition, etc., based on grammar rules and models, prior to recognizing named entities. The identified verb may be used by the IC module 264 to identify intent, which is then used by the NER module 262 to identify frameworks. A framework for an intent of “play” may specify a list of slots/fields applicable to play the identified “object” and any object modifier (e.g., a prepositional phrase), such as {Artist Name}, {Album Name}, {Song name}, etc. The NER module 260 then searches the corresponding fields in the domain-specific and personalized lexicon(s), attempting to match words and phrases in the query tagged as a grammatical object or object modifier with those identified in the database(s).

[0041] This process includes semantic tagging, which is the labeling of a word or combination of words according to their type/semantic meaning. Parsing may be performed using heuristic grammar rules, or an NER model may be constructed using techniques such as hidden Markov models, maximum entropy models, log linear models, conditional random fields (CRF), and the like.

[0042] For instance, a query of “play mother’s little helper by the rolling stones” might be parsed and tagged as {Verb}: “Play,” {Object}: “mother’s little helper,” {Object Preposition}: “by,” and {Object Modifier}: “the rolling stones.” At this point in the process, “Play” is identified as a verb based on a word database associated with the music domain, which the IC module 264 will determine corresponds to the “play music” intent. No determination has been made as to the meaning of “mother’s little helper” and “the rolling stones,” but based on grammar rules and models, it is determined that these phrase relate to the grammatical object of the query.

[0043] The frameworks linked to the intent are then used to determine what database fields should be searched to determine the meaning of these phrases, such as searching a

user’s gazette for similarity with the framework slots. So a framework for “play music intent” might indicate to attempt to resolve the identified object based {Artist Name}, {Album Name}, and {Song name}, and another framework for the same intent might indicate to attempt to resolve the object modifier based on {Artist Name}, and resolve the object based on {Album Name} and {Song Name} linked to the identified {Artist Name}. If the search of the gazetteer does not resolve the a slot/field using gazetteer information, the NER module 262 may search the database of generic words associated with the domain (in the NLU’s knowledge base 272). So for instance, if the query was “play songs by the rolling stones,” after failing to determine an album name or song name called “songs” by “the rolling stones,” the NER 262 may search the domain vocabulary for the word “songs.” In the alternative, generic words may be checked before the gazetteer information, or both may be tried, potentially producing two different results.

[0044] The comparison process used by the NER module 262 may classify (i.e., score) how closely a database entry compares to a tagged query word or phrase, how closely the grammatical structure of the query corresponds to the applied grammatical framework, and based on whether the database indicates a relationship between an entry and information identified to fill other slots of the framework.

[0045] The NER modules 262 may also use contextual operational rules to fill slots. For example, if a user had previously requested to pause a particular song and thereafter requested that the voice-controlled device to “please un-pause my music,” the NER module 262 may apply an inference-based rule to fill a slot associated with the name of the song that the user currently wishes to play—namely the song that was playing at the time that the user requested to pause the music.

[0046] The results of NLU processing may be tagged to attribute meaning to the query. So, for instance, “play mother’s little helper by the rolling stones” might produce a result of: {domain} Music, {intent} Play Music, {artist name} “rolling stones,” {media type} SONG, and {song title} “mother’s little helper.” As another example, “play songs by the rolling stones” might produce: {domain} Music, {intent} Play Music, {artist name} “rolling stones,” and {media type} SONG.

[0047] The output from the NLU processing (which may include tagged text, commands, etc.) may then be sent to a command processor 290, which may be located on a same or separate server 120 as part of system 100. The destination command processor 290 may be determined based on the NLU output. For example, if the NLU output includes a command to play music, the destination command processor 290 may be a music playing application, such as one located on device 110 or in a music playing appliance, configured to execute a music playing command. If the NLU output includes a search query (for example, requesting the return of search results), the destination command processor 290 may include a search engine processor, such as one located on a search server, configured to execute a search command and determine search results, which may include output text to be processed by a TTS engine and output from a device as synthesized speech.

[0048] Typically, an ASR system may be capable of performing speech recognition on speech of various qualities, without specific regard to those certain qualities. For example, an ASR system may be capable of converting an

utterance to text regardless of whether that utterance is whispered, spoken in an excited voice, spoken in a sad voice, whined, shouted, etc. In fact, traditional ASR systems do not care about such voice qualities. Instead, traditional ASR systems only care about recognizing the words in the speech, not any paralinguistic qualities.

[0049] The present system is actually configured to detect speech quality/qualities and determine a label corresponding to the detected qualities that may be applied to an utterance in the speech and used for later processing. The speech quality may be based on paralinguistic metrics that describe some quality/feature other than the specific words spoken. Paralinguistic features may include acoustic features such as speech tone/pitch, rate of change of pitch (first derivative of pitch), speed, prosody/intonation, resonance, energy/volume, hesitation, phrasing, nasality, breath, whether the speech includes a cough, sneeze, laugh or other non-speech articulation (which are commonly ignored by ASR systems), detected background audio/noises, distance between the user and a device, etc.

[0050] Current ASR systems may be configured to detect some such paralinguistic features, however current systems are not configured to analyze those features to put a descriptive label on the speech (such as whisper, etc.) in order to pass that label as an input to downstream processing, such as coordinating the voicing of the input utterance with the voicing of TTS output or execution of a command included in the utterance. The present system includes a speech quality detector, as shown in FIG. 2, that may process paralinguistic feature data to classify one or more qualities of incoming speech and then alter downstream/output operation in response to the one or more qualities.

[0051] For example, based on audio (and possibly non-audio) paralinguistic feature data a system may determine that an input utterance was whispered. Whispered speech is typically “unvoiced,” that is words are spoken using the articulators (mouth, lips, tongue, etc.) as normal, but without use/vibration of vocal cords such that an utterance has no resonance, or resonance below a certain threshold. Vocal resonance is when the product of voicing (i.e., phonation) is enhanced in tone quality (i.e., timbre) and/or intensity by the air-filled cavities through which speech passes on the speech’s way to the outside air. During whispering, air comes through the throat without being modulated by the vocal cords so that what is left is motion of the articulators resulting in a stream of air without valve structure. Whispered speech may also include speech that is at a low volume or volume below a threshold. Some combination of low to no resonance combined with low volume may constitute a whisper for purposes of the system. As noted below, a machine learning model may be trained to recognize whispered speech based on resonance, volume, and/or other features of input audio. While certain spoken whispered sounds may differ from voiced sounds more than others as a result of the lack of voicing or low volume, ASR performance may not necessarily be impacted. That is, current ASR systems may be able to process whispered speech. If ASR performance is impacted, the ASR system may be updated to better recognize whispered speech.

[0052] The system may be configured to recognize that input audio is whispered (which is separate from recognizing the words of whispered speech). For example the system may determine that the input speech has resonance below a threshold and/or a volume below a threshold. Thus the

system may determine that the input speech has an input speech quality corresponding to a whisper/approximated whisper. The system may train components to analyze paralinguistic feature data to make a decision as to whether the speech is whispered. While the system may determine whether speech is whispered based on whether a particular paralinguistic feature value(s) are above a threshold (for example, whether input speech has a resonance under a particular threshold and/or a volume under a particular threshold, etc.), more complex decision making is possible using machine learning models and training techniques. Thus, paralinguistic feature values (whether from audio data or non-audio data) are input as features to a speech quality detector. The speech quality detector may implement a model trained using machine learning techniques to determine a label describing the speech. For example the detector may determine that the speech is whispered. The label (or other indicator of the speech quality) may then be sent to downstream components to alter the output of the device.

[0053] In addition, the system may determine some other speech quality other than whether the speech was whispered. For example, based on the parametric features, the system may determine whether the speaker was speaking in a scoffing or sarcastic tone, the speaker was sniffing or dismissive, the speaker was whining, someone sneezed or coughed, the speaker was talking under his/her breath with others present so only the device will detect the utterance, speech distance, etc.

[0054] The speech quality detector 220 may implement a single model that outputs a label, or may implement a plurality of models, each configured to determine, based on feature values input to the model, whether the speech corresponds to a particular quality. For example one model may be configured to determine whether input speech was whispered, another model may be configured to determine whether input speech was whined, etc. Or, as noted, a single model may be configured to determine multiple labels that may apply to input speech (whisper, whine, shout, etc.) based on that speech’s qualities. The speech quality detector 220 may operate within an ASR sub-system, or as a separate component as part of system 100.

[0055] The system may also consider non-audio data and non-audio features when determining a quality of the speech. For example, if a camera detects the speaker, the system may analyze the video data (for example, the video data may be input to the speech quality detector 220) to determine some quality of the speaker (agitated, subdued, angry, etc.) that the speech quality detector 220 may consider. Other non-audio data may also be input to the speech quality detector 220. For example, time/date data, location data (for example GPS location or relative indoor room location), ambient light data from a light sensor, the identity of other nearby individuals to the speaker, proximity of the user to a device (for example, if a user is leaning in close to a device to speak an utterance, or if a user is far away from the device), etc. The types of acoustic and non-audio data considered by the speech quality detector 220 depends on the types of such data available to the system 100 when processing an utterance. The model(s) available to the speech quality detector 220 may be trained on the various data types available to the speech quality detector 220. For example a first model may be trained to detect that input speech is whispered whereas a second model may be trained to determine that ambient light data from a light sensor is

below a certain threshold. The output from the second model (or more simply, an output from a component such as the light sensor) may indicate to the first model that the atmosphere is dark, which may be used to increase a confidence of the first model that the input speech was whispered. Other such non-audio data may be used to inform a model trained to determine a quality of input speech based on how the non-audio data impacts the classification of the input speech quality.

[0056] Various machine learning techniques may be used to train and/or operate the machine learning models that may be used by the speech quality detector **220**. In machine learning techniques an adaptive system is “trained” by repeatedly providing it examples of data and how the data should be processed using an adaptive model until it can consistently identify how a new example of the data should be processed, even if the new example is different from the examples included in the training set from which it learned. Getting an adaptive model to consistently identify a pattern is in part dependent upon providing the system with training data that represents the desired decision features in such a way that patterns emerge. But provided data with consistent patterns, recognizing such patterns when presented with new and different data is within the capacity of today’s systems, and is in fact used by a wide variety of computer systems ranging from handheld personal consumer electronics to complex massively parallel supercomputers. Such efforts fall into the discipline often referred to as “machine learning,” which is a sub-discipline of artificial intelligence (also known as machine intelligence).

[0057] For example, as above, an adaptive system may be trained using example audio data segments and different values for the various paralinguistic data features available to the system. Different models may be trained to recognize different speech qualities or a single model may be trained to identify applicable speech qualities associated with a particular utterance. For example, a single model may be trained to analyze both audio and non-audio data to determine a speech quality. Alternatively, certain model(s) may be trained to analyze audio data and a separate model(s) may be trained to analyze non-audio data.

[0058] Example machine learning techniques include, for example neural networks, inference engines, trained classifiers, etc. Examples of trained classifiers include support vector machines (SVMs), neural networks, decision trees, AdaBoost (short for “Adaptive Boosting”) combined with decision trees, and random forests. Focusing on SVM as an example, SVM is a supervised learning model with associated learning algorithms that analyze data and recognize patterns in the data, and which are commonly used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. More complex SVM models may be built with the training set identifying more than two categories, with the SVM determining which category is most similar to input data. An SVM model may be mapped so that the examples of the separate categories are divided by clear gaps. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gaps they fall on. Classifiers (either binary or multiple category classifiers) may issue a “score” indicating which category the data most closely

matches. The score may provide an indicator of how closely the data matches the category. For example, in the present application, a support vector machine (SVM) may be trained/configured to process audio data, for example audio feature vectors, to determine if speech associated with the audio feature vectors was whispered. Among the factors the SVM may consider is whether the speech has a resonance below a resonance threshold and/or a volume below a volume threshold. Other features of the speech may also be considered when the SVM classifies the speech as whispered or not-whispered.

[0059] Training a machine learning component such as, in this case, one of the first or second models, requires establishing a “ground truth” for the training examples. In machine learning, the term “ground truth” refers to the accuracy of a training set’s classification for supervised learning techniques. Various techniques may be used to train the models including backpropagation, statistical learning, supervised learning, semi-supervised learning, stochastic learning, or other known techniques. Many different training example utterances may be used to train the models used in the first stage and second stage.

[0060] For example, a model, such as an SVM classifier, may be trained to recognize when an input speech utterance is whispered using many different training utterances, each labeled either “whispered” or “not whispered.” Each training utterance may also be associated with various feature data corresponding to the respective utterance, where the feature data indicates values for the acoustic and/or non-audio paralinguistic features that may be used to determine if a future utterance was whispered. The model may be constructed based on the training utterances and then disseminated to individual devices **110** or to server(s) **120**. A speech quality detector **220** may then use the model(s) to make decisions at runtime as to whether the utterance was whispered. An indicator of the whisper may then be output from the speech quality detector **220** to downstream components such as a command processor **290**, TTS module **314**, etc. The system may then tailor its operations and/or output based on the fact that the utterance was, or was not, whispered. Examples of different models used by the speech quality detector **220** to determine the one or more qualities are shown in FIG. 3 as models **353**.

[0061] Similar training/operation may take place for different speech qualities (excitement, boredom, etc.) where different models are used or a single model is used.

[0062] As shown in FIG. 3, the system may also employ customized models **354** that are customized for particular users. Each user may have multiple such models. The user models **354** may be used by the speech quality detector **220** to select a speech quality in a manner more customized for a specific user. For example, the system may track a user’s utterances to determine how they normally speak, or how they speak under certain conditions, and use that information to train user-specific models **354**. Thus the system may determine the speech quality using some representation of a reference of how a user speaks. The user models **354** may incorporate both audio and non-audio data, which may incorporate not only how a user speaks, but how a user speaks under particular circumstances (i.e., with many individuals present, at different locations, under different lighting conditions, etc. The user models **354** may also take into account eventual commands and/or speech output by the system so that the system may determine how user com-

mands are processed under certain conditions. Each user model 354 may be associated with a user ID, which may be linked to a user profile containing various other information about a particular user. Such profile information may also be used to train the user model 354.

[0063] The speech quality detector 220 may use the models 353, 354 to process audio data 111 and/or non-audio data 302 to determine one or more speech qualities to associate with an input spoken utterance. The speech quality detector 220 may then create an indicator for the determined speech quality/ies. The indicator may then be sent to a downstream command processor 290 so that a command/query may be processed using the indicator and based on the speech quality/ies. The command processor 290 receives the indicator, as well as text and possible other semantic notation related to the utterance, as discussed above in reference to FIG. 2. The command processor 290 may be a component capable of acting on the utterance. Examples of such components include a query processor/search engine, music player, video player, calendaring application, email/messaging application, user interaction controller, personal assistant program, etc. As can be appreciated, many types of command processors 290 are envisioned. The command processor 290 may customize its output based on the speech quality.

[0064] For example, if the command processor 290 is a music player, and the utterance included a request to play music, only did not specify a particular music title, the command processor 290 may use the indicator of speech quality to select a music title. Specifically, if a user shouts, in an excited manner, “PLAY SOME MUSIC!!” the speech quality detector 220 may send an indicator to the command processor that the speech had a quality of excitement and the NLU module 260 may send the command processor 290 text and semantic indicators that the utterance included a request to play music. The command processor 290 may then select a music title to play based on the quality of excitement and may thus select a rock song or similar up-tempo song from a user’s catalog. In another example, if a user whispers “play some music,” the speech quality detector 220 may send an indicator to the command processor that the speech was whispered and the NLU module 260 may send the command processor 290 text and semantic indicators that the utterance included a request to play music. The command processor 290 may then select a music title to play based on the quality of being whispered and may thus select a mellow or calm song from a user’s catalog. Similar selections of actions by different command processors 290 outside the domain of music are also envisioned. As another example, volume of output may be decreased as a result of whispered input speech, or volume increased as a result of excited speech, or the like. As another example, volume of output may be increased if a user is determined to be a long distance away from a device, thus ensuring that the output is loud enough for the user to hear at the user’s distance.

[0065] In another embodiment, the command processor 290 may select a static output based on the speech quality. For example, a system 100, through a command processor 290 or otherwise, may be preconfigured with a number of fixed actions with specific outputs that may be taken in response to specific input speech qualities. Such preconfigured responses may be determined and stored ahead of time, and selected for output based on an input speech quality. Specifically, a static output of a spoken reprimand may be

output in response to speech of a certain quality. Such as “stop whining” if the speech quality detector 220 determines input speech to be whined, “no need to shout” if the speech quality detector 220 determines input speech to be shouted, “ask nicely” if the speech quality detector 220 detects angry speech, “do you need help” if the speech quality detector 220 determines input speech to be in distress, or other examples. The static output may also be selected based on an indication that ASR or NLU processing failed. For example, if the speech quality detector 220 detects the speech to be whispered and ASR and/or NLU processing failed, the system may output a static response of “please do not whisper, I did not understand.” Many other such responses are possible based on the detected speech quality.

[0066] In a further example of customizing output based on input speech quality, a TTS component of the system may be configured to synthesize speech based on a determined speech quality. A TTS module 314 may receive the indicator of input speech quality and may configure an output speech quality (if output speech is called for) to correspond to (or even match or approximate) the input speech quality. For example, if a user whispers an utterance including a query to a device 110, the device may send the audio to a server 120. The server may process the audio with a speech quality detector 220 to determine the utterance was whispered and to send an indicator that the speech was whispered to the TTS module 314. The server (or another server) may perform ASR and NLU processing to identify text associated with the query. A command processor 290 may then process the text to determine a textual answer responding to the query. The textual answer may be sent to the TTS module 314 so the TTS module 314 may synthesize speech corresponding to the textual answer. However the TTS module 314 may, based on the indicator, synthesize whispered speech (or speech configured to approximate a whisper) to output to the user. In a broader example, the TTS module 314 may synthesize speech based on one or more speech qualities of the input speech as detected by the speech quality detector 220. Speech may be synthesized by the TTS module as described below.

[0067] The TTS module/processor 314 includes a TTS front end (TTSFE) 316, a speech synthesis engine 318, and TTS storage 320. The TTSFE 316 transforms input text data (for example from command processor 290) into a symbolic linguistic representation for processing by the speech synthesis engine 318. The speech synthesis engine 318 compares the annotated phonetic units models and information stored in the TTS storage 320 for converting the input text into speech. The TTSFE 316 and speech synthesis engine 318 may include their own controller(s)/processor(s) and memory or they may use the controller/processor and memory 310 of the server 120, device 110, or other device, for example. Similarly, the instructions for operating the TTSFE 316 and speech synthesis engine 318 may be located within the TTS module 314, within the memory and/or storage of the server 120, device 110, or within an external device.

[0068] Text input into a TTS module 314 may be sent to the TTSFE 316 for processing. The front-end may include modules for performing text normalization, linguistic analysis, and linguistic prosody generation. During text normalization, the TTSFE processes the text input and generates standard text, converting such things as numbers, abbrevia-

tions (such as Apt., St., etc.), symbols (\$, %, etc.) into the equivalent of written out words.

[0069] During linguistic analysis the TTSFE 316 analyzes the language in the normalized text to generate a sequence of phonetic units corresponding to the input text. This process may be referred to as phonetic transcription. Phonetic units include symbolic representations of sound units to be eventually combined and output by the system as speech. Various sound units may be used for dividing text for purposes of speech synthesis. A TTS module 314 may process speech based on phonemes (individual sounds), half-phonemes, di-phones (the last half of one phoneme coupled with the first half of the adjacent phoneme), bi-phones (two consecutive phonemes), syllables, words, phrases, sentences, or other units. Each word may be mapped to one or more phonetic units. Such mapping may be performed using a language dictionary stored by the system, for example in the TTS storage module 320. The linguistic analysis performed by the TTSFE 316 may also identify different grammatical components such as prefixes, suffixes, phrases, punctuation, syntactic boundaries, or the like. Such grammatical components may be used by the TTS module 314 to craft a natural sounding audio waveform output. The language dictionary may also include letter-to-sound rules and other tools that may be used to pronounce previously unidentified words or letter combinations that may be encountered by the TTS module 314. Generally, the more information included in the language dictionary, the higher quality the speech output.

[0070] Based on the linguistic analysis the TTSFE 316 may then perform linguistic prosody generation where the phonetic units are annotated with desired prosodic characteristics, also called acoustic features, which indicate how the desired phonetic units are to be pronounced in the eventual output speech. During this stage the TTSFE 316 may consider and incorporate any prosodic annotations that accompanied the text input to the TTS module 314. Such acoustic features may include pitch, energy, duration, and the like. Application of acoustic features may be based on prosodic models available to the TTS module 314. Such prosodic models indicate how specific phonetic units are to be pronounced in certain circumstances. A prosodic model may consider, for example, a phoneme's position in a syllable, a syllable's position in a word, a word's position in a sentence or phrase, neighboring phonetic units, etc. As with the language dictionary, prosodic model with more information may result in higher quality speech output than prosodic models with less information. Further, a prosodic model and/or phonetic units may be used to indicate particular speech qualities of the speech to be synthesized, where those speech qualities may match the speech qualities of input speech (for example, the phonetic units may indicate prosodic characteristics to make the ultimately synthesized speech sound like a whisper based on the input speech being whispered).

[0071] The output of the TTSFE 316, referred to as a symbolic linguistic representation, may include a sequence of phonetic units annotated with prosodic characteristics. This symbolic linguistic representation may be sent to a speech synthesis engine 318, also known as a synthesizer, for conversion into an audio waveform of speech for output to an audio output device 204 and eventually to a user. The speech synthesis engine 318 may be configured to convert the input text into high-quality natural-sounding speech in

an efficient manner. Such high-quality speech may be configured to sound as much like a human speaker as possible, or may be configured to be understandable to a listener without attempts to mimic a precise human voice.

[0072] A speech synthesis engine 318 may perform speech synthesis using one or more different methods. In one method of synthesis called unit selection, described further below, a unit selection engine 330 matches the symbolic linguistic representation created by the TTSFE 316 against a database of recorded speech, such as a database of a voice corpus. The unit selection engine 330 matches the symbolic linguistic representation against spoken audio units in the database. Matching units are selected and concatenated together to form a speech output. Each unit includes an audio waveform corresponding with a phonetic unit, such as a short .wav file of the specific sound, along with a description of the various acoustic features associated with the .wav file (such as its pitch, energy, etc.), as well as other information, such as where the phonetic unit appears in a word, sentence, or phrase, the neighboring phonetic units, etc. Using all the information in the unit database, a unit selection engine 330 may match units to the input text to create a natural sounding waveform. The unit database may include multiple examples of phonetic units to provide the system with many different options for concatenating units into speech. One benefit of unit selection is that, depending on the size of the database, a natural sounding speech output may be generated. As described above, the larger the unit database of the voice corpus, the more likely the system will be able to construct natural sounding speech.

[0073] In another method of synthesis called parametric synthesis parameters such as frequency, volume, noise, are varied by a parametric synthesis engine 332, digital signal processor or other audio generation device to create an artificial speech waveform output. Parametric synthesis uses a computerized voice generator, sometimes called a vocoder. Parametric synthesis may use an acoustic model and various statistical techniques to match a symbolic linguistic representation with desired output speech parameters. Parametric synthesis may include the ability to be accurate at high processing speeds, as well as the ability to process speech without large databases associated with unit selection, but also typically produces an output speech quality that may not match that of unit selection. Unit selection and parametric techniques may be performed individually or combined together and/or combined with other synthesis techniques to produce speech audio output.

[0074] Parametric speech synthesis may be performed as follows. A TTS module 314 may include an acoustic model, or other models, which may convert a symbolic linguistic representation into a synthetic acoustic waveform of the text input based on audio signal manipulation. The acoustic model includes rules which may be used by the parametric synthesis engine 332 to assign specific audio waveform parameters to input phonetic units and/or prosodic annotations. The rules may be used to calculate a score representing a likelihood that a particular audio output parameter(s) (such as frequency, volume, etc.) corresponds to the portion of the input symbolic linguistic representation from the TTSFE 316.

[0075] The parametric synthesis engine 332 may use a number of techniques to match speech to be synthesized with input phonetic units and/or prosodic annotations. One common technique is using Hidden Markov Models

(HMMs). HMMs may be used to determine probabilities that audio output should match textual input. HMMs may be used to translate from parameters from the linguistic and acoustic space to the parameters to be used by a vocoder (the digital voice encoder) to artificially synthesize the desired speech. Using HMMs, a number of states are presented, in which the states together represent one or more potential acoustic parameters to be output to the vocoder and each state is associated with a model, such as a Gaussian mixture model. Transitions between states may also have an associated probability, representing a likelihood that a current state may be reached from a previous state. Sounds to be output may be represented as paths between states of the HMM and multiple paths may represent multiple possible audio matches for the same input text. Each portion of text may be represented by multiple potential states corresponding to different known pronunciations of phonemes and their parts (such as the phoneme identity, stress, accent, position, etc.). An initial determination of a probability of a potential phoneme may be associated with one state. As new text is processed by the speech synthesis engine 318, the state may change or stay the same, based on the processing of the new text. For example, the pronunciation of a previously processed word might change based on later processed words. A Viterbi algorithm may be used to find the most likely sequence of states based on the processed text. The HMMs may generate speech in parametrized form including parameters such as fundamental frequency (f_0), noise envelope, spectral envelope, etc. that are translated by a vocoder into audio segments. The output parameters may be configured for particular vocoders such as a STRAIGHT vocoder, TANDEM-STRAIGHT vocoder, HNM (harmonic plus noise) based vocoders, CELP (code-excited linear prediction) vocoders, GlottHMM vocoders, HSM (harmonic/stochastic model) vocoders, or others.

[0076] An example of HMM processing for speech synthesis is shown in FIG. 4. A sample input phonetic unit, for example, phoneme /E/, may be processed by a parametric synthesis engine 332. The parametric synthesis engine 332 may initially assign a probability that the proper audio output associated with that phoneme is represented by state S_0 in the Hidden Markov Model illustrated in FIG. 4. After further processing, the speech synthesis engine 318 determines whether the state should either remain the same, or change to a new state. For example, whether the state should remain the same 404 may depend on the corresponding transition probability (written as $P(S_0|S_0)$), meaning the probability of going from state S_0 to S_0 and how well the subsequent frame matches states S_0 and S_1 . If state S_1 is the most probable, the calculations move to state S_1 and continue from there. For subsequent phonetic units, the speech synthesis engine 318 similarly determines whether the state should remain at S_1 , using the transition probability represented by $P(S_1|S_1)$ 408, or move to the next state, using the transition probability $P(S_2|S_1)$ 410. As the processing continues, the parametric synthesis engine 332 continues calculating such probabilities including the probability 412 of remaining in state S_2 or the probability of moving from a state of illustrated phoneme /E/ to a state of another phoneme. After processing the phonetic units and acoustic features for state S_2 , the speech recognition may move to the next phonetic unit in the input text.

[0077] The probabilities and states may be calculated using a number of techniques. For example, probabilities for

each state may be calculated using a Gaussian model, Gaussian mixture model, or other technique based on the feature vectors and the contents of the TTS storage 320. Techniques such as maximum likelihood estimation (MLE) may be used to estimate the probability of particular states.

[0078] In addition to calculating potential states for one audio waveform as a potential match to a phonetic unit, the parametric synthesis engine 332 may also calculate potential states for other potential audio outputs (such as various ways of pronouncing phoneme /E/) as potential acoustic matches for the phonetic unit. In this manner multiple states and state transition probabilities may be calculated.

[0079] The probable states and probable state transitions calculated by the parametric synthesis engine 332 may lead to a number of potential audio output sequences. Based on the acoustic model and other potential models, the potential audio output sequences may be scored according to a confidence level of the parametric synthesis engine 332. The highest scoring audio output sequence, including a stream of parameters to be synthesized, may be chosen and digital signal processing may be performed by a vocoder or similar component to create an audio output including synthesized speech waveforms corresponding to the parameters of the highest scoring audio output sequence and, if the proper sequence was selected, also corresponding to the input text.

[0080] Unit selection speech synthesis may be performed as follows. Unit selection includes a two-step process. First a unit selection engine 330 determines what speech units to use and then it combines them so that the particular combined units match the desired phonemes and acoustic features and create the desired speech output. Units may be selected based on a cost function which represents how well particular units fit the speech segments to be synthesized. The cost function may represent a combination of different costs representing different aspects of how well a particular speech unit may work for a particular speech segment. For example, a target cost indicates how well a given speech unit matches the features of a desired speech output (e.g., pitch, prosody, etc.). A join cost represents how well a speech unit matches a consecutive speech unit for purposes of concatenating the speech units together in the eventual synthesized speech. The overall cost function is a combination of target cost, join cost, and other costs that may be determined by the unit selection engine 330. As part of unit selection, the unit selection engine 330 chooses the speech unit with the lowest overall combined cost. For example, a speech unit with a very low target cost may not necessarily be selected if its join cost is high.

[0081] The system may be configured with one or more voice corpuses for unit selection. Each voice corpus may include a speech unit database. The speech unit database may be stored in TTS storage 320, in storage 312, or in another storage component. For example, different unit selection databases may be stored in TTS voice unit storage 372. Each speech unit database includes recorded speech utterances with the utterances' corresponding text aligned to the utterances. A speech unit database may include many hours of recorded speech (in the form of audio waveforms, feature vectors, or other formats), which may occupy a significant amount of storage. The unit samples in the speech unit database may be classified in a variety of ways including by phonetic unit (phoneme, diphone, word, etc.), linguistic prosodic label, acoustic feature sequence, speaker identity, etc. The sample utterances may be used to create

mathematical models corresponding to desired audio output for particular speech units. When matching a symbolic linguistic representation the speech synthesis engine **318** may attempt to select a unit in the speech unit database that most closely matches the input text (including both phonetic units and prosodic annotations). Generally the larger the voice corpus/speech unit database the better the speech synthesis may be achieved by virtue of the greater number of unit samples that may be selected to form the precise desired speech output. An example of how unit selection is performed is illustrated in FIGS. **5A** and **5B**.

[0082] For example, as shown in FIG. **5A**, a target sequence of phonetic units **502** to synthesize the word “hello” is determined by a TTS device. As illustrated, the phonetic units **502** are individual phonemes, though other units, such as diphones, etc. may be used. A number of candidate units **504** may be stored in the voice corpus. Although phonemes are illustrated in FIG. **5A**, other phonetic units, such as diphones, may be selected and used for unit selection speech synthesis. For each phonetic unit there are a number of potential candidate units (represented by columns **506**, **508**, **510**, **512** and **514**) available. Each candidate unit represents a particular recording of the phonetic unit with a particular associated set of acoustic and linguistic features. The TTS system then creates a graph of potential sequences of candidate units to synthesize the available speech. The size of this graph may be variable based on certain device settings. An example of this graph is shown in FIG. **5B**. A number of potential paths through the graph are illustrated by the different dotted lines connecting the candidate units. A Viterbi algorithm may be used to determine potential paths through the graph. Each path may be given a score incorporating both how well the candidate units match the target units (with a high score representing a low target cost of the candidate units) and how well the candidate units concatenate together in an eventual synthesized sequence (with a high score representing a low join cost of those respective candidate units). The TTS system may select the sequence that has the lowest overall cost (represented by a combination of target costs and join costs) or may choose a sequence based on customized functions for target cost, join cost or other factors. The candidate units along the selected path through the graph may then be combined together to form an output audio waveform representing the speech of the input text. For example, in FIG. **5B** the selected path is represented by the solid line. Thus units #₂, H₁, E₄, L₃, O₃, and #₄ may be selected, and their respective audio concatenated, to synthesize audio for the word “hello.”

[0083] Audio waveforms including the speech output from the TTS module **314** may be sent to an audio output component, such as a speaker for playback to a user or may be sent for transmission to another device, such as another server **120**, for further processing or output to a user. Audio waveforms including the speech may be sent in a number of different formats such as a series of feature vectors, uncompressed audio data, or compressed audio data. For example, audio speech output may be encoded and/or compressed by an encoder/decoder (not shown) prior to transmission. The encoder/decoder may be customized for encoding and decoding speech data, such as digitized audio data, feature vectors, etc. The encoder/decoder may also encode non-TTS data of the system, for example using a general encoding scheme such as .zip, etc.

[0084] A TTS module **314** may be configured to perform TTS processing in multiple languages. For each language, the TTS module **314** may include specially configured data, instructions and/or components to synthesize speech in the desired language(s). To improve performance, the TTS module **314** may revise/update the contents of the TTS storage **320** based on feedback of the results of TTS processing, thus enabling the TTS module **314** to improve speech recognition.

[0085] Other information may also be stored in the TTS storage **320** for use in speech recognition. The contents of the TTS storage **320** may be prepared for general TTS use or may be customized to include sounds and words that are likely to be used in a particular application. For example, for TTS processing by a global positioning system (GPS) device, the TTS storage **320** may include customized speech specific to location and navigation. In certain instances the TTS storage **320** may be customized for an individual user based on his/her individualized desired speech output. For example a user may prefer a speech output voice to be a specific gender, have a specific accent, speak at a specific speed, have a distinct emotive quality (e.g., a happy voice), or other customizable characteristic(s) (such as approximating whispering) as explained in other sections herein. The speech synthesis engine **318** may include specialized databases or models to account for such user preferences.

[0086] For example, to create the customized speech output of the system, the system may be configured with multiple voice corpuses/unit databases **378a-378n**, where each unit database is configured with a different “voice” to match desired speech qualities. The voice selected by the TTS module **314** to synthesize the speech. For example, one voice corpus may be stored to be used to synthesize whispered speech (or speech approximating whispered speech), another may be stored to be used to synthesize excited speech (or speech approximating excited speech), and so on. To create the different voice corpuses a multitude of TTS training utterance may be spoken by an individual and recorded by the system. The TTS training utterances used to train a TTS voice corpus may be different from the training utterances used to train an ASR system or the models used by the speech quality detector. The audio associated with the TTS training utterances may then be split into small audio segments and stored as part of a voice corpus. The individual speaking the TTS training utterances may speak in different voice qualities to create the customized voice corpuses, for example the individual may whisper the training utterances, say them in an excited voice, and so on. Thus the audio of each customized voice corpus may match the respective desired speech quality. The customized voice corpuses **378** may then be used during runtime to perform unit selection to synthesize speech having a speech quality corresponding to the input speech quality.

[0087] Additionally, parametric synthesis may be used to synthesize speech with the desired speech quality. For parametric synthesis, parametric features may be configured that match the desired speech quality. For example, if simulated whispered speech was desired parametric features may indicate the resulting speech should have a low volume and a low resonance (i.e., simulate be “unvoiced”). If simulated excited speech was desired, parametric features may indicate an increased speech rate and/or pitch for the resulting speech. Many other examples are possible. The desired parametric features for particular speech qualities

may be stored in a “voice” profile and used for speech synthesis when the specific speech quality is desired. Customized voices may be created based on multiple desired speech qualities combined (for both unit selection or parametric synthesis). For example, one voice may be “whispered” while another voice may be “whispered and excited.” Many such combinations are possible.

[0088] As an alternative to customized voice corpuses or customized parametric “voices,” one or more filters may be used to alter traditional TTS output to match the desired one or more speech qualities. For example, a TTS module **314** may synthesize speech as normal, but the system (either as part of the TTS module **314** or otherwise) may apply a filter to make the synthesized speech sound take on the desired speech quality. Using whispering as an example, if simulated whispered speech is desired a whisper filter may be applied to, for example, remove voice resonance and excitation and add in white noise to approximate whispering. In this manner a traditional TTS output may be altered to take on the desired speech quality.

[0089] During runtime a TTS module **314** may receive text for speech synthesis along with an indicator for a desired speech quality of the output speech, for example, an indicator created by speech quality detector **220**. The TTS module **314** may then select a voice matching the speech quality, either for unit selection or parametric synthesis, and synthesize speech using the received text and speech quality indicator.

[0090] FIG. 6A illustrates a flow diagram describing operation of the system according to various embodiments. The system may receive (**602**) input audio and may determine (**604**) a speech quality of input audio using trained model(s). The system may determine (**606**) an indicator of speech quality and may send (**608**) the indicator to a command processor. The system may also perform (**610**) ASR processing on the input audio to determine utterance text. The system may also perform NLU or further processing on the utterance text. The system may send (**612**) the utterance text, semantic notes, and/or other data to a command processor for execution. The system may then execute (**614**) a command associated with the utterance using the utterance text and the indicator of speech quality, thus customizing the output based on the input speech.

[0091] FIG. 6B illustrates a flow diagram further describing operation of the system according to various embodiments. The system determines (**616**) text for synthesis associated with an executed command and/or input utterance. The system sends (**618**) an indicator of speech quality to a TTS module. The system then selects (**620**) a voice/parametric factors based on the speech quality and synthesizes (**622**) speech using the determined text and selected voice/parametric factors. The system then (**624**) outputs the synthesized speech.

[0092] FIG. 7 is a block diagram conceptually illustrating a local device **110** that may be used with the described system and may incorporate certain speech receiving/keyword spotting capabilities. FIG. 8 is a block diagram conceptually illustrating example components of a remote device, such as a remote server **120** that may assist with ASR, NLU processing, or command processing. Server **120** may also assist in determining similarity between ASR hypothesis results as described above. Multiple such servers **120** may be included in the system, such as one server **120** for ASR, one server **120** for NLU, etc. In operation, each of

these devices may include computer-readable and computer-executable instructions that reside on the respective device (**110/120**), as will be discussed further below.

[0093] Each of these devices (**110/120**) may include one or more controllers/processors (**704/804**), that may each include a central processing unit (CPU) for processing data and computer-readable instructions, and a memory (**706/806**) for storing data and instructions of the respective device. The memories (**706/806**) may individually include volatile random access memory (RAM), non-volatile read only memory (ROM), non-volatile magnetoresistive (MRAM) and/or other types of memory. Each device may also include a data storage component (**708/808**), for storing data and controller/processor-executable instructions. Each data storage component may individually include one or more non-volatile storage types such as magnetic storage, optical storage, solid-state storage, etc. Each device may also be connected to removable or external non-volatile memory and/or storage (such as a removable memory card, memory key drive, networked storage, etc.) through respective input/output device interfaces (**702/802**). The storage component **708/808** may include storage for various data including ASR models **252**, NLU knowledge base **272**, entity library **282**, speech quality models **352**, TTS voice unit storage **372**, or other storage used to operate the system.

[0094] Computer instructions for operating each device (**110/120**) and its various components may be executed by the respective device’s controller(s)/processor(s) (**704/804**), using the memory (**706/806**) as temporary “working” storage at runtime. A device’s computer instructions may be stored in a non-transitory manner in non-volatile memory (**706/806**), storage (**708/808**), or an external device(s). Alternatively, some or all of the executable instructions may be embedded in hardware or firmware on the respective device in addition to or instead of software.

[0095] Each device (**110/120**) includes input/output device interfaces (**702/802**). A variety of components may be connected through the input/output device interfaces, as will be discussed further below. Additionally, each device (**110/120**) may include an address/data bus (**724/824**) for conveying data among components of the respective device. Each component within a device (**110/120**) may also be directly connected to other components in addition to (or instead of) being connected to other components across the bus (**724/824**).

[0096] Referring to the device **110** of FIG. 7, the input/output device interfaces **702** connect to a variety of components such as an audio output component such as a speaker **760**, a wired headset or a wireless headset (not illustrated) or an audio capture component. The audio capture component may be, for example, a microphone **104** or array of microphones, a wired headset or a wireless headset (not illustrated), etc. If an array of microphones is included, approximate distance to a sound’s point of origin may be performed acoustic localization based on time and amplitude differences between sounds captured by different microphones of the array. The microphone **104** may be configured to capture speech including an utterance. The device **110** (using microphone **104**, ASR module **250**, etc.) may be configured to determine audio data corresponding to the utterance. The device **110** (using input/output device interfaces **702**, antenna **714**, etc.) may also be configured to transmit the audio data to server **120** for further processing.

[0097] For example, via the antenna(s), the input/output device interfaces 702 may connect to one or more networks 199 via a wireless local area network (WLAN) (such as WiFi) radio, Bluetooth, and/or wireless network radio, such as a radio capable of communication with a wireless communication network such as a Long Term Evolution (LTE) network, WiMAX network, 3G network, etc. A wired connection such as Ethernet may also be supported. Through the network(s) 199, the speech processing system may be distributed across a networked environment.

[0098] The device 110 and/or server 120 may include an ASR module 250. The ASR module in device 110 may be of limited or extended capabilities. The ASR module 250 may include the language models 254 stored in ASR model storage component 252, and an ASR module 250 that performs the automatic speech recognition process. If limited speech recognition is included, the ASR module 250 may be configured to identify a limited number of words, such as wakewords detected by the device, whereas extended speech recognition may be configured to recognize a much larger range of words.

[0099] The device 110 and/or server 120 may include a limited or extended NLU module 260. The NLU module in device 110 may be of limited or extended capabilities. The NLU module 260 may comprising the name entity recognition module 262, the intent classification module 264 and/or other components. The NLU module 260 may also include a stored knowledge base 272 and/or entity library 282, or those storages may be separately located.

[0100] One or more servers 120 may also include a command processor 290 that is configured to execute commands associate with an ASR hypothesis as described above. One or more servers 120 may also include a machine learning training component 870 that is configured to determine one or more models used by, for example, a speech quality detector 220.

[0101] The device 110 and/or server 120 may include a speech quality detector 220, which may be a separate component or may be included in an ASR module 250. The speech quality detector 220 receives audio data and potentially non-audio data and classifies an utterance included in the audio according to detected qualities of the audio as described above. As described above, the speech quality detector 220 may employ classifier(s) or other machine learning trained models to determine whether qualities associated with an utterance.

[0102] As noted above, multiple devices may be employed in a single speech processing system. In such a multi-device system, each of the devices may include different components for performing different aspects of the speech processing. The multiple devices may include overlapping components. The components of the devices 110 and server 120, as illustrated in FIGS. 7 and 8, are exemplary, and may be located a stand-alone device or may be included, in whole or in part, as a component of a larger device or system.

[0103] As illustrated in FIG. 9, multiple devices (110a-f, 120, 902, 904, and/or 906) may contain components of the system 100 and the devices may be connected over a network 199. The network 199 is representative of any type of communication network, including data and/or voice network, and may be implemented using wired infrastructure (e.g., cable, CATS, fiber optic cable, etc.), a wireless infrastructure (e.g., WiFi, RF, cellular, microwave, satellite, Bluetooth, etc.), and/or other connection technologies.

Devices may thus be connected to the network 199 through either wired or wireless connections. Network 199 may include a local or private network or may include a wide network such as the internet. For example, devices 110, networked camera(s) 902 (which may also include one or more microphones), networked microphone(s) 904 (or networked microphone array(s), not illustrated), networked speaker(s) 906, etc. may be connected to the network 199 through a wireless service provider, over a WiFi or cellular network connection or the like. Other devices, such as server(s) 120, may connect to the network 199 through a wired connection or wireless connection. Networked devices 110 may capture audio using one-or-more built-in or connected microphones 104/904 or audio capture devices, with processing performed by speech quality detector 220, ASR, NLU, or other components of the same device or another device connected via network 199, such as speech quality detector 220, ASR 250, NLU 260, etc. of one or more servers 120c. Further, inputs from camera(s) 902, microphones 904, speaker(s) 906, or other components may be used by the system to provide paralinguistic metrics as described above.

[0104] The concepts disclosed herein may be applied within a number of different devices and computer systems, including, for example, general-purpose computing systems, speech processing systems, and distributed computing environments.

[0105] The above aspects of the present disclosure are meant to be illustrative. They were chosen to explain the principles and application of the disclosure and are not intended to be exhaustive or to limit the disclosure. Many modifications and variations of the disclosed aspects may be apparent to those of skill in the art. Persons having ordinary skill in the field of computers and speech processing should recognize that components and process steps described herein may be interchangeable with other components or steps, or combinations of components or steps, and still achieve the benefits and advantages of the present disclosure. Moreover, it should be apparent to one skilled in the art, that the disclosure may be practiced without some or all of the specific details and steps disclosed herein.

[0106] Aspects of the disclosed system may be implemented as a computer method or as an article of manufacture such as a memory device or non-transitory computer readable storage medium. The computer readable storage medium may be readable by a computer and may comprise instructions for causing a computer or other device to perform processes described in the present disclosure. The computer readable storage media may be implemented by a volatile computer memory, non-volatile computer memory, hard drive, solid-state memory, flash drive, removable disk and/or other media. In addition, components of one or more of the modules and engines may be implemented as in firmware or hardware, such as the acoustic front end 256, which comprise among other things, analog and/or digital filters (e.g., filters configured as firmware to a digital signal processor (DSP)).

[0107] As used in this disclosure, the term “a” or “one” may include one or more items unless specifically stated otherwise. Further, the phrase “based on” is intended to mean “based at least in part on” unless specifically stated otherwise.

1. A computer-implemented method for processing a whispered utterance and responding in whispered synthesized speech, the method comprising:

- receiving input audio data comprising an input utterance;
- processing the input audio data with at least one trained model to determine that the input utterance was whispered;
- performing automatic speech recognition (ASR) on the input audio data to determine input text corresponding to the input utterance;
- performing natural language understanding processing on the input text to identify a query;
- determining content responding to the query based on the input utterance being whispered; and
- causing the content to be output.

2. The computer-implemented method of claim 1, further comprising:

- performing text-to-speech (TTS) processing on output text based on a speech quality indicator to generate output audio data, wherein the output audio data comprises synthesized speech responding to the query, wherein the synthesized speech is configured to sound like a whispered voice, wherein performing TTS processing further comprises:
- performing unit selection using a voice corpus to select a plurality of stored audio data segments of recorded whispered speech, the stored audio data segments corresponding to the output text; and
- concatenating the plurality of stored audio segments to determine the output audio data.

3. The computer-implemented method of claim 1, wherein the trained model comprises a support vector machine (SVM) configured to process audio feature vectors to determine that speech associated with the audio feature vectors has a resonance below a resonance threshold and has a volume below a volume threshold.

4. A computer-implemented method comprising:

- determining an input speech quality corresponding to input audio data;
- performing automatic speech recognition on the input audio data to determine input text;
- determining content based on the input text and the input speech quality; and
- causing the content to be output.

5. The computer-implemented method of claim 4, wherein determining the input speech quality comprises processing the input audio data using at least one trained classifier configured to classify the audio data as either corresponding to the speech quality or not corresponding to the speech quality.

6. The computer-implemented method of claim 4, further comprising:

- performing natural language understanding processing on the input text to identify a search query; and
- processing the query with a search engine to obtain a search result;
- wherein determining the content comprises selecting, based on the input speech quality, a portion of the search result as the content.

7. The computer-implemented method of claim 4, further comprising determining the input speech quality indicates that the audio data corresponds to whispered speech.

8. The computer-implemented method of claim 7, wherein determining the input speech quality comprises

processing the input audio data with a trained classifier configured to process audio feature vectors to determine that the input audio data has a resonance below a resonance threshold and has a volume below a volume threshold.

9. The computer-implemented method of claim 8, further comprising processing input non-audio data to determine the input speech quality.

10. The computer-implemented method of claim 9, wherein processing the input non-audio data comprises:

- receiving light data from a light sensor;
- determining that the light data is below a light threshold; and
- inputting an indication that the light data is below the light threshold into the trained classifier.

11. The computer-implemented method of claim 7, further comprising:

- performing text-to-speech (TTS) processing on output text to generate output audio data, wherein the TTS processing is based on the input speech quality, and wherein performing TTS processing further comprises:
- performing unit selection using a voice corpus to select a plurality of stored audio data segments of recorded whispered speech, the stored audio data segments corresponding to the output text; and
- concatenating the plurality of stored audio segments to determine the output audio data, wherein the output audio data corresponds to an output utterance that responds to the query in a whispered voice.

12. The computer-implemented method of claim 11, further comprising selecting the output text from a plurality of prepared text samples based on the speech quality.

13. A computing system comprising:

- at least one processor;
- a memory including instructions operable to be executed by the at least one processor to cause the system to perform a set of actions comprising:
- determining an input speech quality corresponding to input audio data;
- performing automatic speech recognition on the input audio data to determine input text;
- determining content based on the input text and the input speech quality; and
- causing the content to be output.

14. The computing system of claim 13, wherein determining the input speech quality comprises processing the input audio data using at least one trained classifier configured to classify the audio data as either corresponding to the speech quality or not corresponding to the speech quality.

15. The computing system of claim 13, the set of actions further comprising:

- performing natural language understanding processing on the input text to identify a search query; and
- processing the query with a search engine to obtain a search result;
- wherein determining the content comprises selecting, based on the input speech quality, a portion of the search result as the content.

16. The computing system of claim 13, the set of actions further comprising determining the input speech quality indicates that the audio data corresponds to whispered speech.

17. The computing system of claim 16, wherein determining the input speech quality comprises processing the input audio data with a trained classifier configured to process audio feature vectors to determine that the input audio data has a resonance below a resonance threshold and has a volume below a volume threshold.

18. The computing system of claim 17, the set of actions further comprising processing input non-audio data to determine the input speech quality.

19. The computing system of claim 18, wherein processing the input non-audio data comprises:

- receiving light data from a light sensor;
- determining that the light data is below a light threshold;
- and

- inputting an indication that the light data is below the light threshold into the trained classifier.

20. The computing system of claim 16, the set of actions further comprising:

- performing text-to-speech (TTS) processing on output text to generate output audio data, wherein the TTS processing is based on the input speech quality, and wherein performing TTS processing further comprises:
- performing unit selection using a voice corpus to select a plurality of stored audio data segments of recorded whispered speech, the stored audio data segments corresponding to the output text; and

- concatenating the plurality of stored audio segments to determine the output audio data, wherein the output audio data corresponds to an output utterance that responds to the query in a whispered voice.

21. The computing system of claim 20, the set of actions further comprising selecting the output text from a plurality of prepared text samples based on the speech quality.

22. The computer-implemented method of claim 4, further comprising:

- performing natural language understanding processing on the input text to identify a query;
- determining first content and second content that are responsive to the query; and
- selecting the first content as the content for output based on the input speech quality.

23. The computer-implemented method of claim 4, further comprising:

- performing natural language understanding processing on the input text to determine the input text corresponds to a request to play music;
- determining first music content and second music content that are responsive to the request; and
- determining the first music content includes an audio quality corresponding to the input speech quality; and
- selecting the first music content as the content for output.

* * * * *