JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY (JKUAT)

DEPARTMENT OF COMPUTING

ICS 2406: COMPUTER SYSTEMS PROJECT

LITERATURE REVIEW REPORT

TITLE: AN IMPLEMENTATION OF SPEECH RECOGNITION IN AN EMERGENCY APPLICATION

AUTHOR:

NAME:	NGIGI SHALON NJERI	REG. NO.:	CS281-1566/2014
SUBMISSION DATE:	_____	SIGNATURE:	_____
COURSE:	B. SC. COMPUTER SCIENCE		

SUPERVISOR 1:

NAME:	PROF. WAWERU MWANGI	SIGNATURE:	_____
DATE:	_____		

SUPERVISOR 2:

NAME:	_____	SIGNATURE:	_____
DATE:	_____		

OVERVIEW

The aim of this project is to design and implement a speech recognition android-based application, purposed for use during emergency situations. It should allow a user to interact with it through voice commands, using preset keywords to activate it, and to navigate through and seek assistance. For example, voice commands to enable calling an operator or send an automatic text to a listed emergency contact.

The aim of this phase however, is to detail research on Speech Recognition from published literature.

OBJECTIVES

1. To understand the process of speech recognition and its fundamentals
2. To study various ways of implementing speech recognition
3. To compare various tools for implementing speech recognition

Table of Contents

INTRODUCTION

Speech recognition is the inter-disciplinary sub-field of computational linguistics, that develops methodologies and technologies which enable the recognition and translation of spoken language into text by computers, also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT). [1]

Speech recognition is the task of comparing an unknown speaker with a set of known speakers, achieved by breaking down sounds the hardware detects into smaller non-divisible sounds called phonemes, and attempting to match the detected phonemes with known words from a stored dictionary. The speech recognition process can be thought of as having a front end and a back end. [2]

The front end converts sound signal to digital format and processes the audio stream, isolating segments of sound that are probably speech and converting them into a series of numeric values that characterize the vocal sounds in the signal.

The back end is a specialized search engine that takes the output produced by the front end and searches across three databases: an acoustic model, a lexicon, and a language model.

- The *acoustic model* represents acoustic sounds of a language, and can be trained to recognize the characteristics of a particular user's speech patterns and acoustic environments
- The *lexicon* lists a large number of words in the language, and provides information on how to pronounce each word.
- The *language model* represents the ways in which the words of a language are combined

A speech recognition engine or speech recognizer takes an audio stream as input and turns it into a text transcription. The quality of a recognizer is determined by how good it is at refining its search, eliminating the poor matches, and selecting the more likely matches. This depends largely on the quality of its language and acoustic models and the effectiveness of its algorithms, both for processing sound and for searching across the models.

CHALLENGES OF SPEECH RECOGNITION

Speech recognition software has seen great advancement since it was first invented, but it still faces several problems that restrict it from being used exclusively as a method of transcription. These problems include variations in the pronunciation of words, unwanted ambient noises, homonyms and individual accents.

- **Noisy and reverberant environments** cause speech recognition systems to perform poorly. It can be difficult to separate words from the background noise.
- **Continuous speech.** When a speaker speaks quickly and run all their words in a long stream, it is difficult to establish where a word ends and where another begins.
- **Speaker vocal characteristics**. Everyone's voice is a little different, accounting for accents and differences in pronunciation of words; an individual's voice also changes from moment to moment. This makes modeling a speaker independent system challenging.
- **Homonyms.** These are words that sound identical but mean extremely different things, e.g. red (color) and read (past tense of the act of perusing a book). They are hard to recognize accurately and in context.
- **Overlapping speeches.** A challenge in the process, is to understand the speech uttered by different users, current systems have a difficulty to separate simultaneous speeches form multiple users.

APPLICATIONS OF SPEECH RECOGNITION [2]

- **Voice biometrics.** This technology compares the previously stored voice print or template with the utterance and produces score. Biometrics reduce each spoken word into some frequency segments called formants. This technology is used at various agencies like online banking, online security trading, online information services, computer access security and many more.
- **Siri technology.** Application is used by Apple in their iPhones. It captures voice from the speaker and performs actions narrated by the speaker e.g. calling a specific person on your contact list.

- **From a medical perspective.** People with disabilities can benefit from speech recognition programs. Speech recognition is especially useful for people who have difficulty using their hands, in such cases speech recognition programs are much beneficial and they can use for operating computers. Speech recognition is used in deaf telephony, such as voicemail to text.
- **Games and toys.** Various voice driven toys and games are available in the market.
- **Fighter aircrafts.** The fighter jets are controlled with the voice system and are given commands from the base.
- **Home automation.** Voice activated alarms and control signals help provide security at homes

THE FUTURE OF SPEECH RECOGNITION

- Accuracy will become better and better.
- Dictation speech recognition will gradually become accepted.
- Greater use will be made of “intelligent systems” which will attempt to guess what the speaker intended to say, rather than what was actually said, as people often misspeak and make unintentional mistakes.
- Microphone and sound systems will be designed to adapt more quickly to changing background noise levels, different environments, with better recognition of extraneous material to be discarded.

TYPES OF SPEECH RECOGNITION

Speech recognition systems can be divided into the number of classes based on their ability to recognize words and list of words they have. A few classes of speech recognition are classified as under:

a. **Isolated Speech**

   Isolated words usually involve a pause between two utterances. A system accepts single words or single utterances at a time.

b. **Connected Speech**

   Connected words or connected speech is similar to isolated speech but allow separate utterances with minimal pause between them.

c. **Continuous speech**

   Continuous speech allow the user to speak almost naturally, it is also called the computer dictation.

d. **Spontaneous Speech**

   At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters.

1. THE PROCESS OF SPEECH RECOGNITION

Sounds are captured by a sensor, such as a microphone, converted to digital format and divided into segments (frames). The sound signals (frames) are turned into a sequence of numbers (vectors) representing pressure change over time using speech processing techniques. The ASR system converts this time-pressure signal into a time-frequency-energy signal. Feature vectors extracted are decoded and characteristics are compared to a dictionary of phonemes (testing and matching), and the closest match is selected.

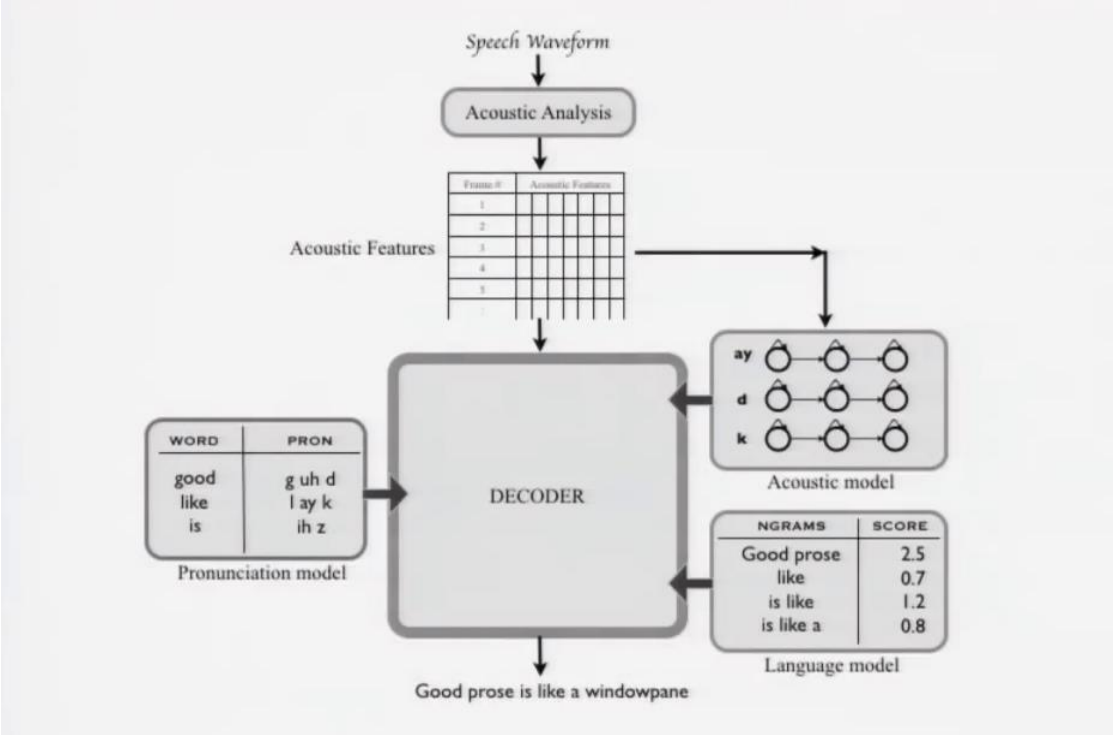


Figure: ASR Process

The ASR system has been trained on a curated set of labelled speech sounds, and so labels the sounds it is presented with. These acoustic labels are combined with a model of word pronunciation and a model of word sequences, to create a textual representation of what was said. The goal of speech recognition is to analyze, extract, characterize and recognize information that shows the speaker identity.

1.1 FRONT-END ASR SYSTEM

In order for recognition to take place, there is a need to convert the analog speech signal not only into a digital format but also into one in which it is possible to differentiate between the fundamental components of speech.

Front-end ASR is responsible for conversion of analog speech signal to a digital format, environmental noise filtering as well as feature extraction [2].

Steps in front-end ASR:

1.1.1 Speech Analysis.

Sounds are waves of pressure propagated through the air, the sounds are captured by a sensor such as a microphone. The first step involves converting the sound from its analog form to digital format, a process called **digitization.** It involves sampling and quantization processes. Sampling is converting a continuous signal into discrete signal, while the process of approximating a continuous range of values is known as quantization.

The digital data is converted into a spectrogram (a graph showing how the component frequencies of the sound change in intensity over time) using a mathematical techniques called a Fast-Fourier Transform (FFT). It is then broken into a series of overlapping chunks called acoustic frames of about 20–30 ms in length. This length is usually chosen because it allows for the short time analysis of the speech signal due to the fact that small segments of the waveform can be considered stationary for time segments of this length. These are then digitally processed in various ways and analyzed to find the components of speech they contain and get signal characteristics such as total energy, zero crossing strength across various frequency ranges.

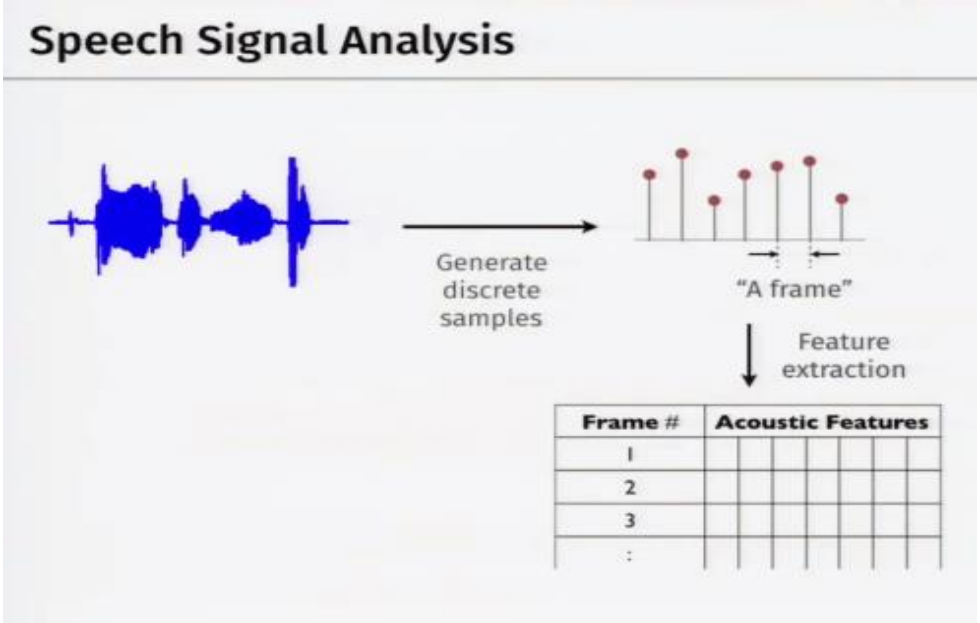


Figure: Signal Analysis (Sampling and quantization)

1.1.2 Speech Feature Extraction Techniques.

In speaker identification the number of training and test vectors needed for the classification problem grows exponentially with the dimensions of the given input, so the need for feature extraction arises.

The primary goal of feature extraction is to simplify recognition by summarizing the vast amount of speech data and obtaining acoustic properties that define speaker individuality. The feature extractor converts the digital speech signal into a sequence of numerical descriptors, called vectors that represent pressure change over time, using speech processing techniques such as Linear Predictive Coding (LPC) and Mel Frequency Cepstral Coefficients (MFCC). The features provide a more stable, robust and compact representation than the raw input signal [3].

Feature extraction can be considered as a data reduction process that attempts to capture the essential characteristics of the speaker with a small data rate. This stage often referred to as speech processing front end [3].

But extracted features should meet some criteria while dealing with the speech signal [3] such as:

- Easy to measure extracted speech features
- It should not be susceptible to mimicry
- It should show little fluctuation from one speaking environment to another
- It should be stable over time
- It should occur frequently and naturally in speech.

Feature extraction techniques

During the feature-detection process, the spectral is converted to a set of features. Among the features are nasality (nasal resonance), frication (random excitation), format locations (frequency of the first three resonances), voiced/ unvoiced classification (periodic or a periodic excitation) and energy ratios.

Initially a spectral and or temporal analysis of the speech signal is performed to give observation vectors which can be used in training. One way to obtain observation vectors from speech samples is to perform spectral analysis. A type of spectral analysis that is often used is linear predictive coding (LPC).

a. Linear Predictive Coding (LPC)

Linear Prediction is a method for using previous information to predict the next value in a sequence. We assume we have a chunk of training information which we can learn the sequence behavior from, then we can apply our learning to situations where the next point is unknown. One of the first things to choose when doing linear prediction is the order of our model: this is the number of previous points we use to predict the next one. If 5 data points are used to predict a sixth, then we call it an order 5 model.

This is a powerful signal analysis technique. The idea behind LPC is that a speech sample can be approximated as a linear combination of past speech samples. Through minimizing the sum of squared differences (over a finite interval) between the actual speech samples and predicted

values, a unique set of parameters or predictor coefficients can be determined. The predictor coefficients are therefore transformed to a more robust set of parameters known as Cepstral coefficients which are then used to obtain feature vectors [4].

b. Mel Frequency Cepstral Coefficients (MFCC)

This method is extensively used in feature extraction in ASR. MFCC mimics the logarithmic perception of loudness and pitch of human auditory system and tries to eliminate speaker dependent characteristics by excluding the fundamental frequency and their harmonics. To represent the dynamic nature of speech the MFCC also includes the change of the feature vector over time as part of the feature vector [7].

Frequency bands are positioned logarithmically in MFCC, it approximates the human system response more loosely than any other system. The technique of computing MFCC is based on the short-term analysis, and thus from each frame a MFCC vector is computed. In order to extract the coefficients, the speech sample is taken as the input and hamming window is applied to minimize the discontinuities of a signal. The Discrete Fourier Transformer (DFT) is used to generate the Mel filter bank. After warping the numbers of coefficients are obtained. Finally the inverse Discrete Fourier Transformer is used for the Cepstral coefficients calculation.

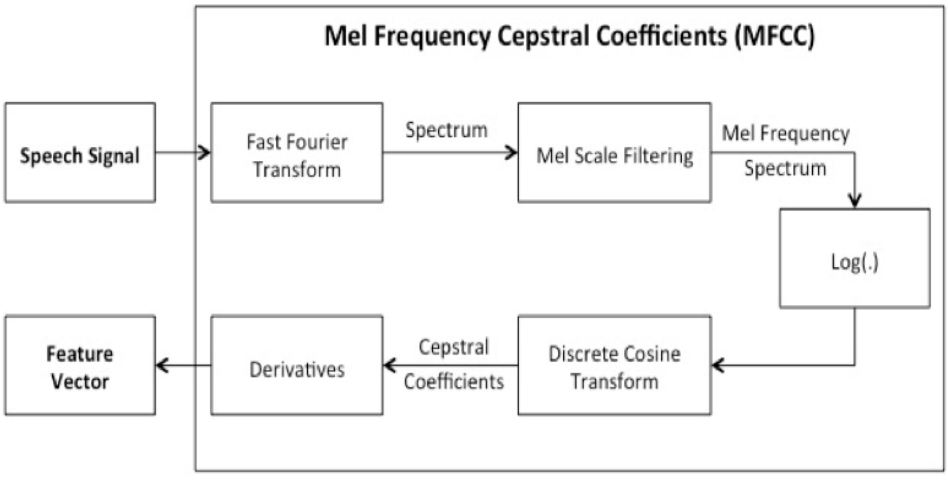


Figure: Block diagram of the MFCC algorithm

1.2 BACK-END OF ASR SYSTEM

The backend of the speech recognition system is responsible for the decoding of the input feature vectors and performing the computations necessary for recognition of the speech pattern initially input into the frontend. It comprises acoustic models for a given vocabulary, a language modelling module and usually a confidence scoring module with their arrangement [2].

STEPS FOR BACK-END ASR SYSTEM

1.2.1 MODELING

A model describes some mathematical function that gathers common attributes of the spoken word. An audio model of a senone (a senone is a phone considered in its wider context), simply put is the most probable feature vector.

Modeling in speech recognition is the process of representing the relationship between an audio signal and linguistic units that make up speech. The acoustic model models the relationship between the audio signal and the phonetic units in the language. The language model is responsible for modelling the word sequences in the language. These two models are combined to get the top-ranked word sequences corresponding to a given audio segment. [8]

An acoustic model and a language model to represent the statistical properties of speech.

a. Acoustic Modeling

It is based on the theory of acoustic phonetics. The theory proposes that there is are finite, distinct phonetics units in spoken language. Each word can be represented as a string of phones. The phonetic units are characterized by a set of properties that are embedded in the speech signal or its spectrum.

The system tries to decode the speech signal in a sequential manner based on the observed acoustic features of the speech waveform and the known relations between acoustic features and phonetic symbols. Coefficients obtained from feature extraction are used as input to an acoustic model along with other features.

Given feature vectors (they describe the acoustic properties of the various phonetic units) obtained from feature extraction process, a recognizer tries to determine the best matching word or sequence of words.

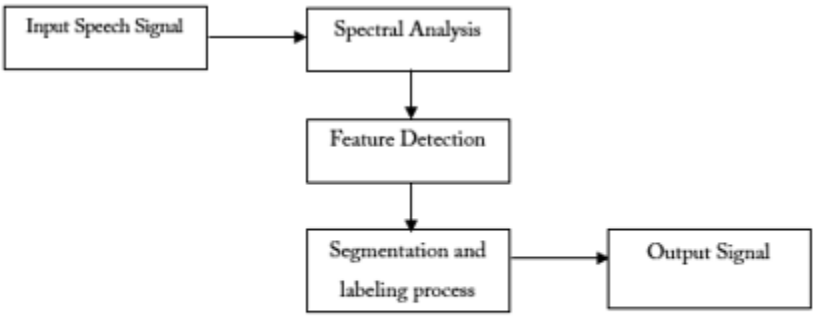
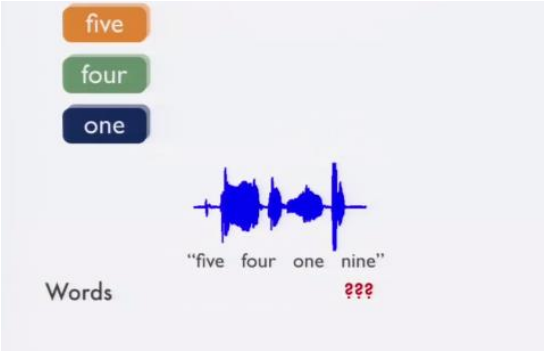


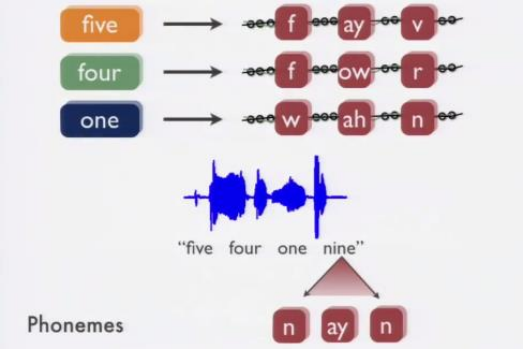
Figure: Acoustic phonetic speech block diagram

An acoustic model is a file that contains statistical representations of each of the distinct sounds that make up a word. Each of these statistical representations is assigned a label called a phoneme.

Segmentation and labelling stage. The system finds the feature stable regions and then labels those regions accordingly in order to match each individual phonetic units.

The result of segmentation and labelling can be viewed in the figure below. Given the string of speech five four, one and nine, where during training the word samples five, four and one have been encountered. Models can be built for five, four and one. What happens when nine is encountered during analysis. When samples are broken down into phonemes, the units of nine can be extrapolated from already present samples.





An acoustic model is created by taking a large database of speech (called a speech corpus) and using special training algorithms to create statistical representations for each phoneme in a language. These statistical representations are called Hidden Markov Models (HMMs). Each phoneme has its own HMM. HMMs are useful in mapping acoustic features to phonemes.

Hidden Markov Models (HMMs)

In this model, the process is described as a sequence of states which change with each other with a certain probability. HMM is a temporal probabilistic model in which the state of the process is described by a single discrete random variable. The possible values of the variable are the possible states of the world [5].

Most current automatic speech recognition systems are based on HMMs.

HMM design is characterized by three fundamental problems:

1. The evaluation of the probability of a sequence of observations given a specific HMM.
2. The determination of a best sequence of model states.
3. The adjustment of model parameters to best account for the observed signal.

Designing HMMs is the problem of determining a method to maximize the probability of the observation sequence given the model.

Hidden Markov Model is a paradigm which is used to learn the mapping of acoustic features to phonemes. Each acoustic feature vectors corresponds to a frame. How would one chunk how many frames correspond to a particular phone? The figure below shows a single chain of phones put together. At each point one could transition to any phone because one wouldn't

know which phone is going to appear next. The entire model is probabilistic. Estimates show for example, these initial ten frames most likely correspond to a certain phone and the transition probabilities determine how many speech frames are going correspond to a particular phone. Once in a particular state there are probabilities for having generated each of the vectors. Therefore one in a particular state there is probability for having seen the vector given that one is in that state.

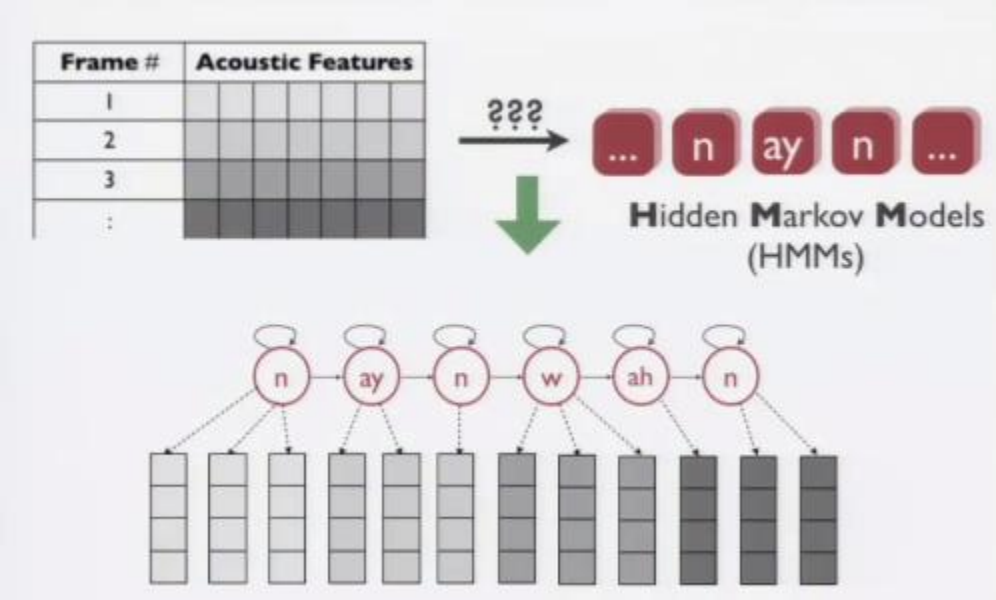
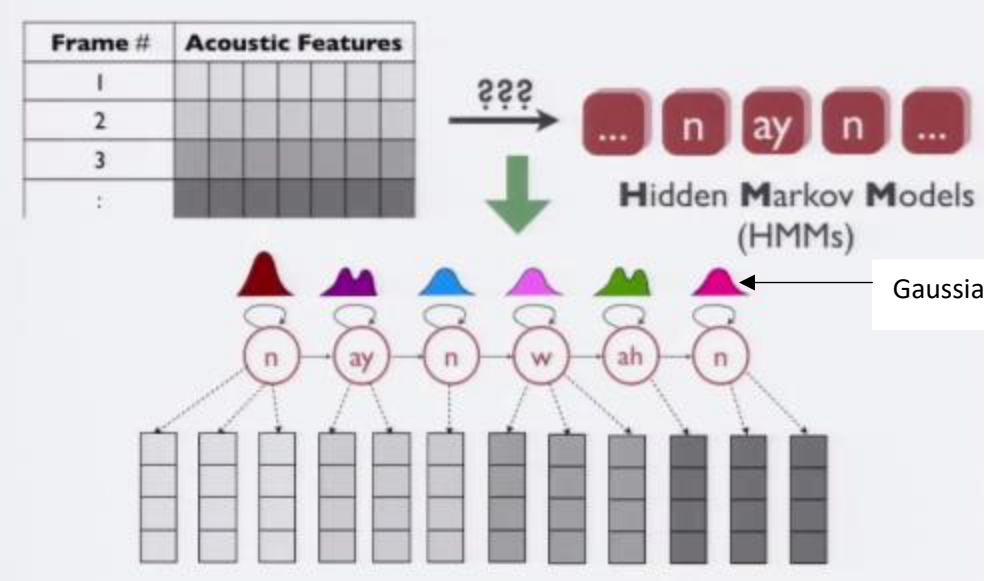


Figure: Map from Acoustic features to phonemes

The probabilities come from Gaussian Mixture Models (GMMs) which is a probability distribution which determines that once one is in a particular state a particular speech vector could be generated with a certain probability. The probabilities are learned from training data.



Types of HMMs

There are many different types of hidden Markov models. In the ergodic or fully connected HMM, every state of the model can be reached (in a single step) from every other state of the model. In speech processing, the left-right model or Bakis model has been used. The benefit of this model is that it can model signals whose properties change over time.

Limitations of HMMs

HMMs have been successfully applied to problems in both isolated and connected word recognition. There are however some limitations of this type of statistical model for speech.

One major limitation is the assumption made in the model that successive observations are independent.

A second limitation is the assumption that the distributions of individual observation parameters can be well represented as a mixture of normal or autoregressive densities.

The final assumption limiting the HMM model is the assumption that the probability of being in a given state at a certain time only depends on the previous state. This assumption is inappropriate as speech sound dependencies often extend through several states [6].

Deep Neural Networks (DNNs):

HMMs have been state of the art models for a long time. Deep Neural Networks can be used for the same mapping. The input to a DNN would be all acoustic features put together and the output would be what the most likely phone to be produced given the set of speech frames. It determines the most likely phone given a speech frame. It is a probability distribution of the phones.

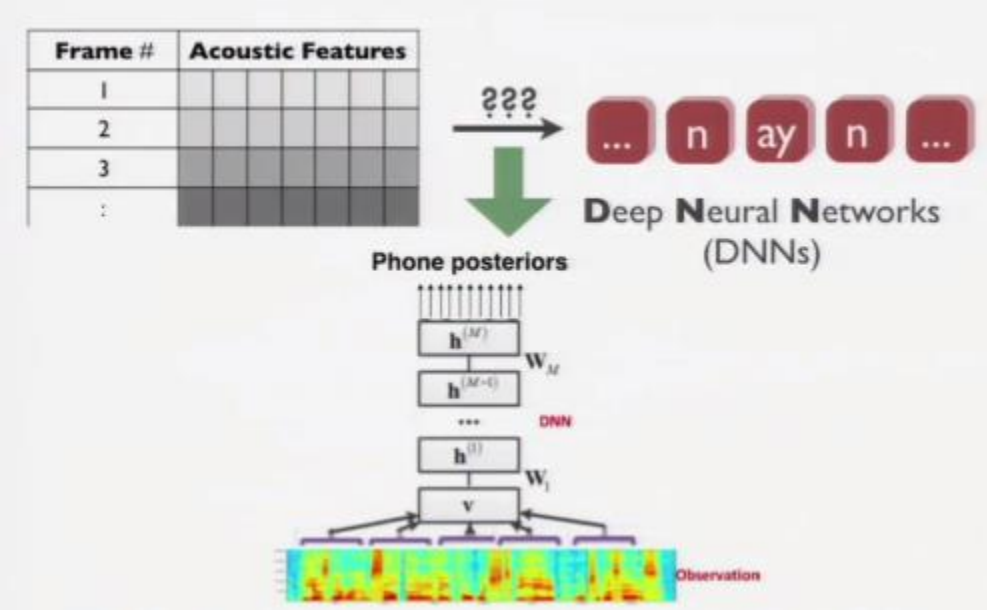


Figure: A map from acoustic features to phonemes

An example (an acoustic model process for a word "house"):

If the system is set up with a simple grammar file to recognize the word "house" (whose phonemes are: "hh aw s"), the steps the speech recognition engine might take are:

The speech decoder listens for distinct sounds spoken and looks for a matching HMM in the acoustic model. When it finds a matching HMM in the acoustic model the decoder takes note of the phoneme. The decoder keeps track of the matching phonemes until it reaches a pause in the user's speech. When a pause is reached, the decoder looks up the matching series of phonemes it has heard in its pronunciation dictionary to determine which word was spoken. In this case it would be:

- HOUSDEN [HOUSDEN] hh aw s d ax n

- **HOUSE [HOUSE] hh aw s (THIS ONE)**
- HOUSE'S [HOUSE'S] hh aw s ix z

The decoder would then look in the grammar file for a matching word or phrase, and since the grammar contains only one word, it would return “house” to the calling program.

Speech recognition engines work best if the acoustic model they use was trained with speech audio which was recorded at the same sampling rate per sample as the speech being recognized.

In summary

The task of the acoustic model is to compute the probability P (O|W), that the pronunciation of the word string W will produce the extracted features [10]. A probabilistic model that maps a sequence of feature vectors to sequence of phones.

Problems and dilemmas faced by Acoustic phonetic approach:

- Need for extensive knowledge of acoustic properties of phonetic units.
- Features are often based on non-optimal ad hoc considerations rather than based on intuition
- The choice of features is likely based on suboptimal and so optimal implementation of Classification and Regression Tree (CART) methods is rarely achieved.
- There is no well-defined, automatic procedure for tuning the labelled speech
- There is no standard way in labelling the training speech

b. Pronunciation Model

This is a model that provides a link between sub-word units (phonemes) and the word. Typically a simple dictionary of pronunciations is maintained, they specify words and their corresponding sequence of phones. This module is not learned from training data, they are given by linguist experts. The output of the pronunciation model is words after mapping a phonetic sequence to a word.

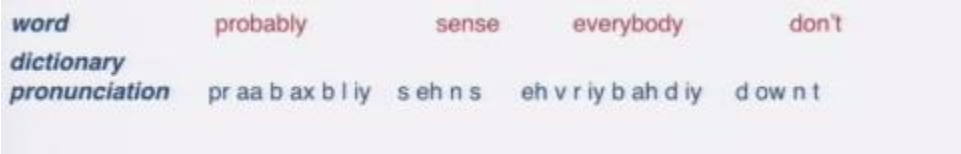


Figure: A word and its subsequent phonetic sequences

c. Language Modeling

A Statistical Language Model is a probability distribution over sequences of words. It is a file used by a Speech Recognition Engine that contains a large list of words and their probability of occurrence. The language model comes in to order the words according to a particular language. It determines how likely it is for different words to occur given the recent word context. The language is trained by a lot of text from a particular language and finds likely occurrences of words. For example, the phrase “the dog” is most likely to be followed by “can” or “ran” but less likely to be followed by the word “pan”.

The language model is also used to disambiguate between similar acoustics. For example the phrase “Let us pray” is more likely compared to “Lettuce spray”.

Language models are used to constrain search in a decoder by limiting the number of possible words that need to be considered at any one point in the search. The consequence is faster execution and higher accuracy.

Language models constrain search either absolutely (by enumerating some small subset of possible expansions) or probabilistically (by computing a likelihood for each possible successor word). The former will usually have an associated grammar this is compiled down into a graph, the latter will be trained from a corpus.

Statistical language models (SLMs) are good for free-form input, such as dictation or spontaneous speech, where it's not practical or possible to a priori specify all possible legal word sequences. [9]

Language models (LMs) assign a probability estimate P (W) to word sequences $W = \{W_1 \ldots W_n\}$. Language models help guide and constrain the search among alternative word hypotheses during recognition.

To estimate word probabilities, counts of words and word context are accumulated over a large text corpora, how often a set of words appear. Relative or normalized counts are computed to get word probabilities.

The language model seeks to refine the output of the pattern matching section by attempting to limit the number of invalid output strings, by placing a confidence score on consecutive output words or phonemes.

N-grams

One of the most commonly used language models is the N-gram model. An n-gram is a contiguous sequence of n items from a given sequence of text. The N-gram language model calculates the probability of a word ending an N word sequence given the last N–1 word [2]. If N being observed is two then it is a bigram, if three a trigram. If one uses π-gram one can reconstruct English sentences very well. Given a sentence, s, we can construct a list of n-grams from s by finding pairs of words that occur next to each other. For example, given the sentence "I am Sam" you can construct bigrams (n-grams of length 2) by finding consecutive pairs of words [11].

E.g.

$\Pr(\text{class} \mid \text{she taught a}) = \frac{\pi(\text{she taught a class})}{\pi(\text{she taught a})}$. Where π ("…") refers to counts derived from a large English text corpus.

This example is a four-gram "she taught a class". The probability that the four-gram follows the particular context "she taught a". Counts of "she taught a class" in large volumes of English text are normalized with the counts of "she taught a", which is the word context, how often the word "class" comes after the context.

The limitation of this model is that we would never see enough data. Even working with counts from large text corpora, there will be many unseen bigrams/ trigrams that never appear in the corpus.

General Strengths and Weaknesses of the Pattern Recognition include:

Weaknesses

- Performance of the system is sensitive to the amount of training data available for creating Sound Class Reference Patterns. Generally the more training, the higher the performance of the system for virtually any task.
- The reference patterns are sensitive to the speaking environment and transmission characteristics of the medium used to create the speech. This is because the speech spectral characteristics are affected by transmission and background noise.
- The computational load for both pattern training and pattern classification is generally linearly proportional to the number of patterns being trained or recognized. Hence, computation of a large number of sound classes could and often does become prohibitive.

Strengths

- No speech-specific knowledge is used explicitly in the system. Hence, the method is relatively insensitive to the choice of vocabulary words, task syntax, and task semantics.
- Because the system is insensitive to sound class, the basic techniques are applicable to a wide range of speech sound, including phrases, whole words, and sub-word units. A basic set of techniques developed for one sound class (e.g. words) can generally be directly applied to different sound classes (e.g., sub-word units) with little or no modifications to the algorithms.
- It is relatively straightforward to incorporate syntactic (and even semantic) constraints directly into the pattern recognition structure, thereby improving recognition accuracy and reducing computation.

1.2.2 FEATURE MATCHING

This step is a search problem finding the most probable word sequence given acoustic observations. Words are composed of state sequences a criterion may be expressed by summing over all the state sequences. The Viterbi criterion: approximate the sum over all sate sequences by using the most probable state sequence.

The task of the search (or decoding) algorithm is to determine the most probable state sequence given the acoustic, pronunciation and language models. In a large vocabulary the task of evaluating all possible word sequences is infeasible.

The decoder searches through the space where all the components (acoustic, pronunciation and language models) are put together.

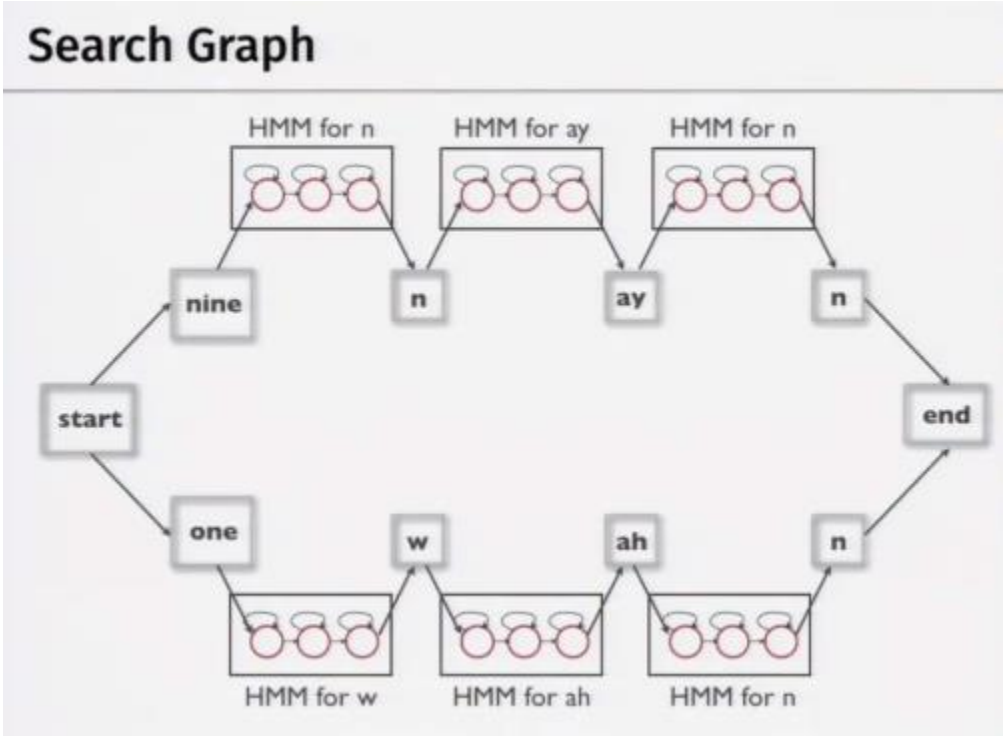


Figure: A simple search graph expecting only two words, say nine and one.

The figure above shows what a naïve search graph would look like. One would start at a particular point and expect two words nine or one. Transitioning into a word, say nine, each of

which is a sequence of phones and each phone corresponds to its own HMM which has its own probabilities.

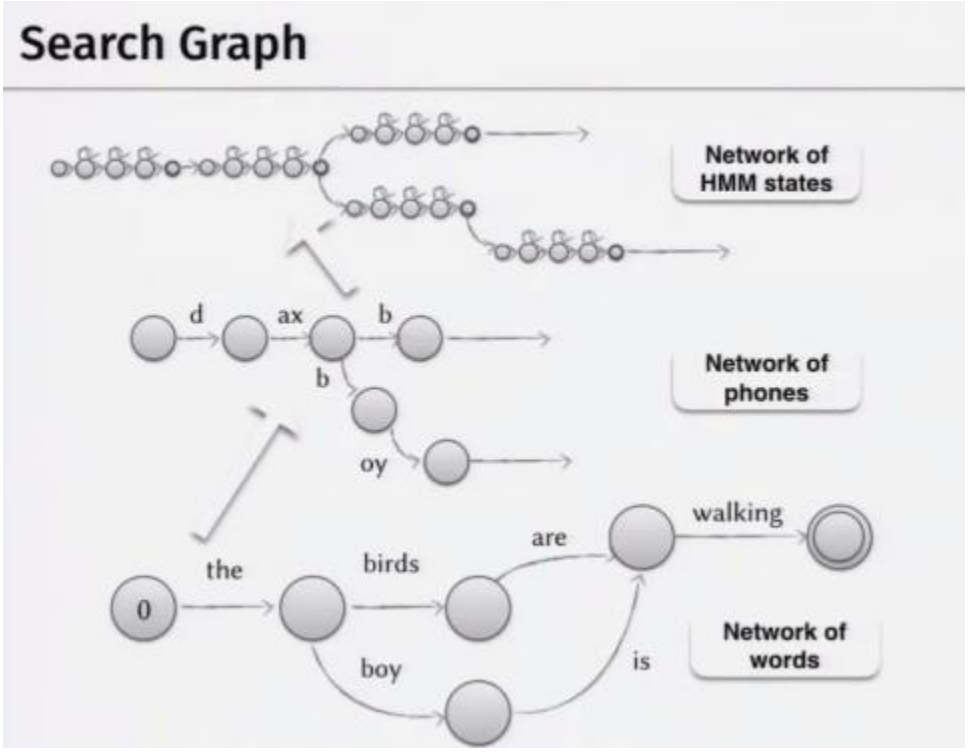


Figure: A compilation of a recognition network. A build of a network of HMMS from a network of phones from a network of words.

Each of the words corresponds to its phone sequence. Then each of the phones correspond to its underlying HMMs.

One would need to search through the graphs to come up with the most likely word sequence that can be obtained from a speech signal. An exact search on the graph is impossible. One would have to result to approximate search techniques.

SEARCH TECHNIQUES:

There are many different pattern matching (or searching) techniques. These include Dynamic Time Warping, Beam Search, Dynamic search structures, Multipass search, Best-first search, Weighted Finite State Transducer (WFST), and Template matching approaches [12].

a. Dynamic Time Warping

DTW algorithm is implemented to calculate least distance between features of word uttered and reference templates. DTW finds the optimal alignment between two times series if one time series may be “warped” non-linearly by stretching or shrinking it along its time axis [14]. This approach finds similarities between two different sequences which vary with time and speech. Speech is considered as a sequence of acoustic vectors and thus different utterances of the same word or sentence will have differing durations for the sounds. In pattern classification, this will lead to a potential mismatch with the stored representations that are developed from training materials. DTW algorithm uses the point matching method to compute the matching distance, the matching distances between test voice and each reference speech template must be computed [15].

In general, DTW is a method that calculates an optimal match between two given sequences (e.g. time series) with certain restrictions [13, Wikipedia].

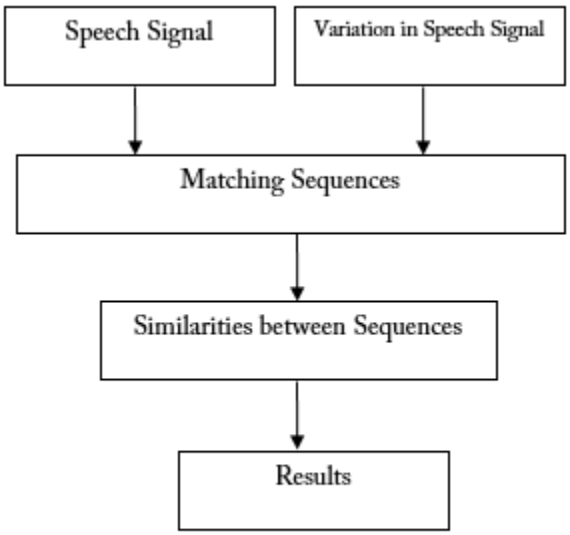


Figure: Dynamic Time Warping Block Diagram

b. Template based approaches

In this approach, a collection of prototypical speech patterns are stored as reference patterns which represents the dictionary of candidate words. An unknown spoken utterance is matched with each of these reference templates and a category of the best matching pattern is selected.

During testing phase each new frame is compared with all of the reference frames and identifies the new utterance as being the word associated with the template with the smallest distance to the new sequence. Usually template for each word is constructed. This has the advantage of using perfectly accurate word models [4].

The Template Matching Approach is subject to the following drawbacks:

- Separate template for each word brings dependency with the lexicon size in opposition to smaller units like phones or phonemes
- Nonlinear time alignment is crucial (inevitable different speaking rates, even for the same speaker, we have different speaking rates for the same word)
- Reliability determines the word boundaries.

c. Beam Search

The basic idea is to prune search paths that are unlikely to succeed. Nodes in the graph whose path probability is more than the factor σ less probable than the best path are removed. Both the language model and acoustic model can contribute to pruning.

d. Multipass Search

Rather than computing the single best hypothesis, the decoder can output alternative hypotheses. An N-best list: a list of the most probable hypothesis. A word graph/ word lattice is built where nodes correspond to time (frame), Arcs correspond to word hypotheses (with associated acoustic and language model probabilities). Multipass search using progressively more detailed models. E.g. using bigram language model on the first pass, and trigram on the second pass. Information is transmitted between passes as word graphs. Later passes rescore word graphs produced by earlier passes.

Decision Rule

This is the last major step in the pattern recognition model. The decision rule chooses which reference pattern most closely matches the unknown test pattern. Two decision rules dominate the variety of approaches applicable, the nearest neighbor rule (NN rule) and the K-nearest neighbor rule (KNN).

The NN rule simply chooses the pattern reference with the smallest average distance score. An ordered list of the recognition candidates can be deduced and further processed by the confidence measure.

The KNN rule is applied when each entity (e.g. a word) is represented by two or more reference templates. The KNN rule computes the average score of each entity over the K best results and chooses the pattern reference with the smallest average. As the NN rule, an ordered list must be computed.

Confidence Measure

As the decision rule for the recognition is based on a distance measure it is possible to introduce a confidence measure to provide a measure of certainty of the detection. The confidence measure is used to implement a rejection task. The aim is to accept the correct decision of the recognizer and to reject the false detection.

2. IMPLEMENTING SPEECH RECOGNITION

There are four different approaches a computer can take to turn speech into spoken words:

1. Simple pattern matching, where each spoken word is recognized in its entirety, without analyzing it.
2. Pattern and feature analysis, where each word is broken into bits and recognized from key features such as the vowels it contains
3. Language modeling and statistical analysis, where a knowledge of grammar and the probability of certain words or sounds following on from one another is used to speed up recognition and improve accuracy
4. End-to-end Neural Systems (Artificial neural networks), these are brain-like computer models that can reliably recognize patterns, such as word sounds, after exhaustive training.

2.1 Simple pattern matching

The system simply distinguishes between different sound patterns form a short list of words such as a list of ten numbers, zero through nine, or bleeping sounds from a touch tone phone keypad. The system doesn't do anything complex such as parsing (splitting a string of spoken words into separate words and figuring out their structure), understanding or having knowledge of the syntax (language structure) or semantics (meaning).

Computationally, there's not a huge difference between recognizing phone tones and spoken numbers "zero", "one" or "two", but in each case, the system solves the problem by comparing an entire chunk of sound to similar stored patterns in its memory. There can be a variety in how different people say the words "three" or "four" (speaking in a different tone, speed of speech, background noise) but the ten numbers are sufficiently different from one another for this not to present a huge computational challenge.

Simple pattern matching is usually applied in automatic call centers, where one dials a number, waits for a recorded voice to answer, then either key in or speak one's account number before

pressing more keys to select options, and if the system can't figure out what one is saying, it is easy enough for the call to be transferred to a human operator.

Voice activated dialing is also simple pattern matching, where one simply trains the phone to recognize the spoken version of a name in the phonebook. When one says a name, the phone doesn't do any particularly sophisticated analysis, it simply compares the sound pattern with the ones stored previously and picks the best match.

The systems generally work very reliably because they have small vocabularies, however they are very limiting and would be inefficient for larger vocabularies since a speaker would have to read each of the words three or four times into a microphone until the computer generalized the sound pattern into something it could recognize reliably.

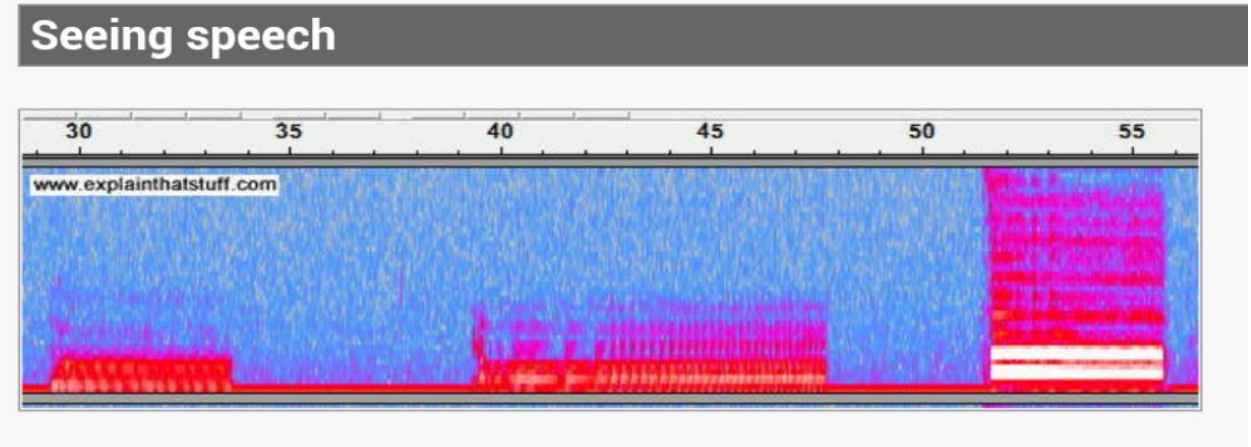
2.2 Pattern and feature analysis

Relatively large vocabularies are made from both common words such as "a", "the" and "but", and other less common ones such as "crepuscular". These words are built from the same basic set of sounds. A computer would store a prototype (a generalized concept of the sounds, from analysis of their key features or components) from which words can be recognized.

The recognition process

Speech recognition systems start by listening to a chunk of sound read through a microphone. The first step involves converting the sound from its analog form to digital format by a piece of hardware or software called an analog-to-digital converter. The digital data is converted into a spectrogram (a graph showing how the component frequencies of the sound change in intensity over time) using a mathematical techniques called a Fast-Fourier Transform (FFT). It is then broken into a series of overlapping chunks called acoustic frames, each typically lasting $1/25^{th}$ to $1/50^{th}$ of a second. These are digitally processed in various ways and analyzed to find the components of speech they contain.

Once utterance is separated into words, and key features of each one have been identified, they are compared with a phonetic dictionary (a list of known words and the sound fragments of features from which they are made) and what was said can probably be identified.



Speech recognition programs start by turning utterances into a spectrogram, which is a three dimensional graph:

- with time showing on the horizontal axis flowing from right to left
- frequency on the vertical axis running from bottom to top and
- color of the chart at each point shows how much energy there is in each frequency of the sound at a given moment

The spectrogram above shows three tones sung into a microphone, each one lasting 5-10 seconds, with a bit of silence in between. First one, is trace of quiet, low frequency sound (dark colors, red and purple, concentrated at the bottom). The second tone is a similar tone to the first but a bit louder (colors appear brighter). The third one has both a higher frequency and intensity (trace goes higher up, higher frequencies, and colors are brighter (more energy)). With training and practice, one can recognize what is being said by looking at a diagram like this.

Theoretically, spoken languages are built form a few dozen phonemes, English uses about 46, while Spanish about 24. One could recognize any spoken utterance just by learning to pick out the phones (or similar key features of spoken languages such as formants – which are

prominent frequencies that can be used to help identify vowels). Instead of having to recognize the sounds of 40,000 words, one would need to recognize the 46 basic component sounds. The method of analyzing spoken words by identifying phones or phonemes is called **beads-on-a-string model:** a chunk of unknown speech is recognized by breaking it down into phones or bits of phones, once you figure out the phones, you can figure out the words.

Most speech recognition systems get better as you use them because they learn as they are being used, using feedback given to them either deliberately or by default. An example of such a system is Dragons dictation system.

The disadvantage with this model is that it assumes speaker independence, when in reality speech is usually very variable.

2.3 Language modelling and Statistical analysis

Speech recognition is however more complex than simply identifying phones and comparing them to stored patterns:

- Speech is extremely variable
- One doesn't always pronounce a certain word the same way
- As a speaker's vocabulary grows, the number of similar sounding words grows. The digits "zero" to "nine" sound different but "zero" sounds like "hero", "one" sounds like "none", "two" like "to, too…"
- The more speakers a system has to recognize, the more variability it is going to encounter and the bigger the likelihood of making mistakes

Language modelling is a higher level of recognition. A language model is used to understand how language really works. When people speak it is unlikely that they are muttering random words and words uttered largely depend on those that come before or after, e.g. "example" is much more likely to be follow words like "for", "an", "better", "good", "bad" than words like "octopus", "table" or even "example". Using rules of grammar it is unlikely that a noun would be spoken before another noun and adjectives would come before nouns and not after them.

So a computer uses rules of grammar and the probability of likely pair of words to make intelligent guesses.

Modern speech recognition systems use complex statistical methods to identify speech. The probability of one phone following another, the probability of silence occurring in between phones, and the likelihood of different words following other words are factored in. Systems build a ***Hidden Markov Model (HMM)*** of each speech segment.

2.4 End-to-end Neural Systems

HMMs have dominated speech recognition, but they are not the only technique used for speech recognition.

In the 1980s computer scientists developed "connectionist" computer models that could mimic how the brain learns to recognize patterns which became known as ***Artificial Neural Networks (ANNs)***

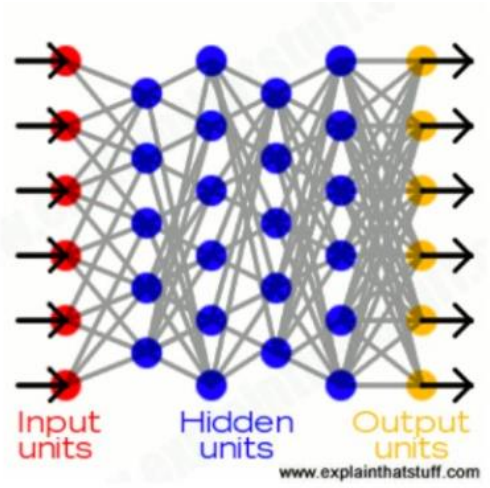
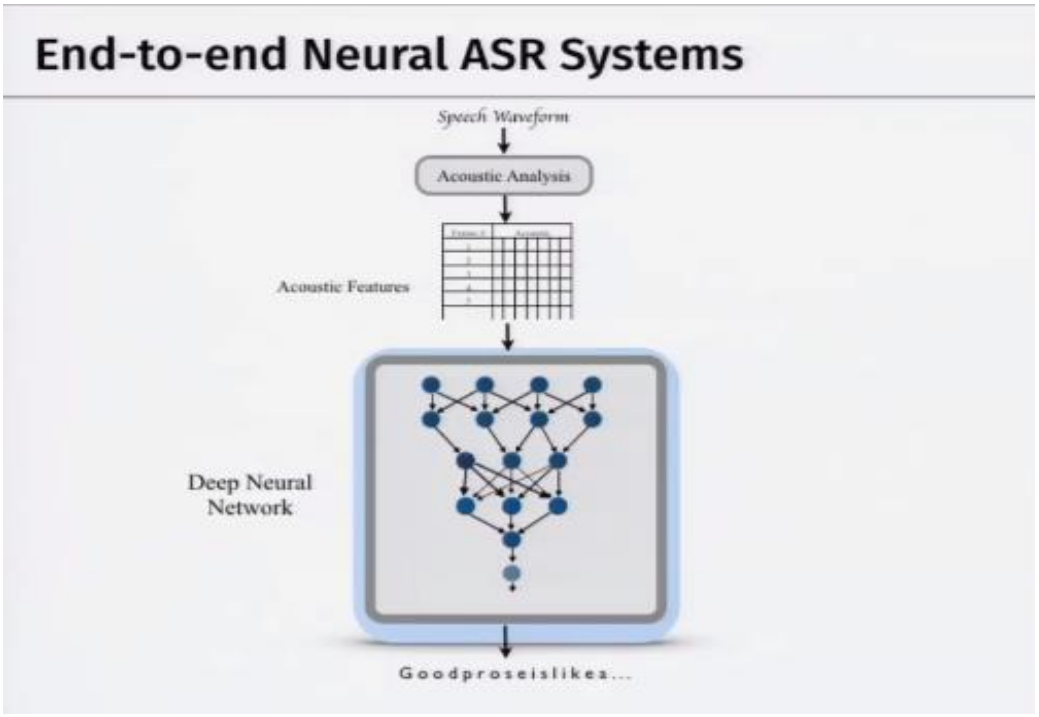


Figure: An Artificial Neural Network

Neural networks are simplified, computerized versions of the brain, which have inputs, outputs and hidden units connecting the two. If trained with enough examples, they learn by gradually adjusting the strength of the connections between the different layers of units. Once a neural

network is fully trained, one can show it an unknown example, and it will attempt to recognize what it is based on the examples it has seen before.

The model seeks to rid of the components and not worry about how a word splits into its corresponding phone sequence but rather learn a mapping directly from acoustic features to letters/ characters. Directly move from speech vectors to a character sequence.



An advantage of this model is that one no longer needs the word to phone mapping which changes for each language. However, for this to work one would need a lot of data. This model is relatively new and still under research.

3. TOOLS FOR SPEECH RECOGNITION

A speech recognition application will typically perform the following basic operations:

1. Initialize the speech recognizer
2. Set the input for speech recognition
3. Create a speech recognition grammar
4. Load the grammar into the speech recognizer
5. Register for speech recognition event notification
6. Create a handler for speech recognition event
7. Start recognition

Note: A recognizer is an installed Runtime Language for speech recognition. A Runtime Language includes the language model, acoustic model, and other data necessary to a speech recognition engine to perform speech recognition in a particular language.

Some tools useful in implementing speech recognition include: CMUSphinx tools, HTK, Kaldi, Simon, Praat, Dragon dictate, and Julius.

3.1 Kaldi

Kaldi is a speech recognition toolkit, freely available under the Apache License. Kaldi is intended for use by speech recognition researchers.

Kaldi aims to provide software that is flexible and extensible. It supports linear transforms, MMI, boosted MMI and MCE discriminative training, feature-space discriminative training, and deep neural networks [20]. Kaldi is similar in aims and scope to HTK

3.2 HTK

Hidden Markov Model Toolkit (HTK) was made for handling HMMs (a statistical parametric synthesis technique). While HTK is mainly used for speech recognition, it can also be used for text-to-speech and for DNA sequencing.

HTK is a portable toolkit for building and manipulating HMMs. It is primarily used for speech recognition research. The tools provide sophisticated facilities for speech analysis, HMM training, testing and results analysis. The software supports HMMs using both continuous

density mixture Gaussians and discrete distributions and can be used to build complex HMM systems [16].

3.3 Simon

Simon is a speech recognition program that provides an easy-to-use user interface. The simple structure and friendly user-interface are some of Simon's biggest strengths. It is an open source speech recognition program that can replace your mouse and keyboard.

Simon uses the KDE libraries, CMU SPHINX and / or Julius coupled with the HTK and runs on Windows and Linux. Simon is known as a popular speech recognition tool for Linux, although it can also work with Windows [18].

3.4 Julius

Julius is a two-pass large vocabulary continuous speech recognition (LVCSR) decoder software for speech-related researchers and developers. Julius continues to be developed by the Interactive Speech Technology Consortium.

It is based on word N-gram and context-dependent HMM. It can perform almost real-time decoding on most current PCs in 60k word dictation task. Major search techniques are fully incorporated such as tree lexicon, N-gram factoring, cross-word context dependency handling, enveloped beam search, Gaussian pruning, Gaussian selection, etc. Besides search efficiency, it is also modularized carefully to be independent from model structures, and various HMM types are supported such as shared-state triphones and tied-mixture models, with any number of mixtures, states, or phones [17].

The main platform is Linux and other UNIX workstations, and also works on Windows. Most recent version is developed on Linux and Windows (Cygwin / mingw), and also has Microsoft SAPI version.

Currently, Japanese is the only language model that's fully available with Julius. A sample English acoustic model is available, but cannot be used for commercial purposes. The VoxForge-Project is working on creating an English language acoustic model for Julius.

3.5 CMUSphinx

Sphinx is a group of speech recognition systems developed by Carnegie Mellon University. There are several packages, each designed for different tasks and applications. CMUSphinx collects over 20 years of the CMU research.

Sphinx is a continuous-speech, speaker-independent recognition system making use of hidden Markov acoustic models (HMMs) and an n-gram statistical language model [19].

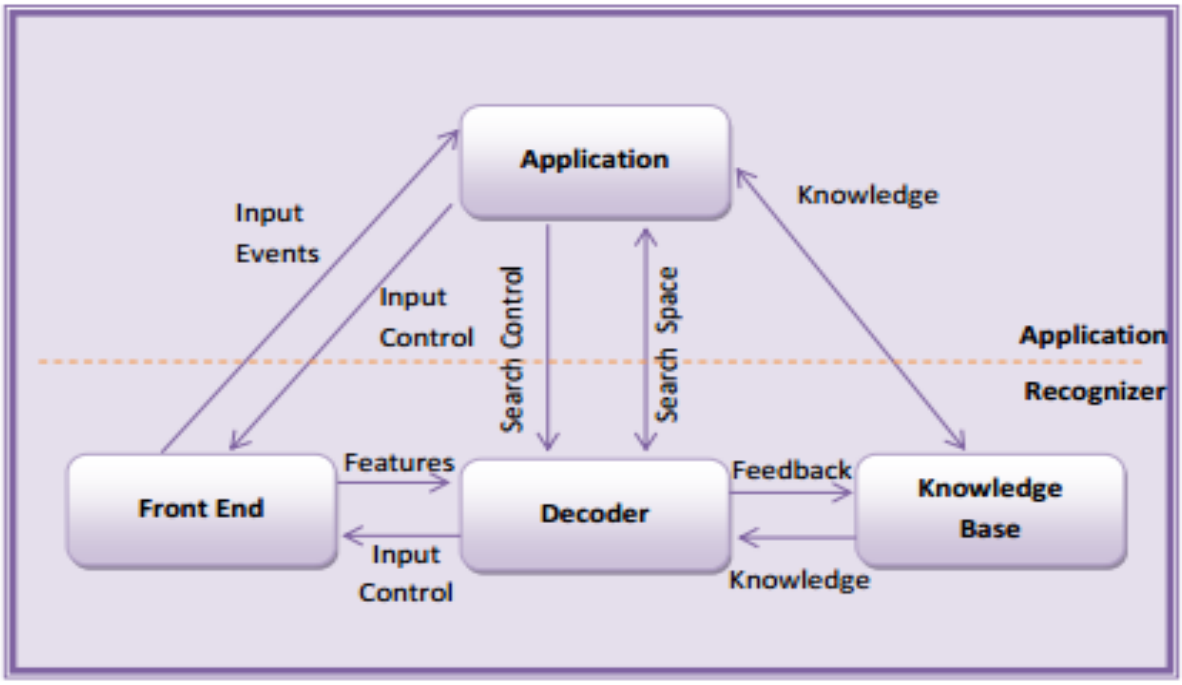


Figure: Architecture of CMUSphinx

CMUSphinx tools are designed specifically for low-resource platforms with a flexible design. The focus is on practical application rather than research. It has support for several languages: US English, UK English, French, Mandarin, German, Dutch, Russian and ability to build models for others. It also has a wide range of tools for many speech recognition related purposes such as keyword spotting, alignment, pronunciation evaluation.

The collection and processing of input data are performed by the front-end. The front-end also performs the feature extraction and passes the results to the decoder. The information regarding acoustic model and language model is passed by the knowledge base to the decoder.

Components managed by the knowledge base are: acoustics model, language model and dictionary.

Acoustic models represent probability distribution of speech sound signals over time where individual series of states represents a single phoneme.

Language models describe the likelihood, probability and penalty taken when a sequence or collection of words is seen. The pronunciations of words are indicated to the decoder using a dictionary file that contains expected sequence of phones for each word.

The decoder receives extracted features of input from front end and combines them with the knowledge base data. It searches the series of features that represents a sequence of words called a search space. It does this by selecting the states, scoring them and generating the result. There is a possibility that more than one state can be selected initially. To generate results, the states have to be ranked according to the required features and then after low scoring states are dropped before generating final result.

Applications of CMUSphinx include voice control, intelligent houses, speech transcription, closed captioning, speech translation, voice search and language learning.

Parts of the toolkit:

- PocketSphinx – A lightweight recognizer library written in C
- Sphinxbase – A support library required by PocketSphinx
- Sphinx 2-4 speech recognizers– An adjustable, modifiable recognizer written in Java
- SphinxTrain – Acoustic model training tools.
- CMUDict – A phonetic dictionary that provides the system with a mapping of vocabulary words to sequences of phonemes. It also contains alternative pronunciations. It supports English, French, German, Russian, Dutch, Italian, Spanish and Mandarin.

Sphinx software systems [22]:

a. Sphinx

A continuous-speech, speaker-independent recognition system making use of HMMs and an n-gram statistical language model. It is of historical interest only it has been superseded in performance by subsequent versions

a. Sphinx2

A fast performance-oriented recognizer. It focuses on real-time recognition suitable for spoken language applications. It incorporates functionality such as end-pointing, partial hypothesis generation, dynamic language model switching and more. It is used in dialog systems and language learning systems. It is no longer under active development

b. Sphinx3

Sphinx2 used a semi-continuous representation for acoustic models. Sphinx3 adopted the prevalent continuous HMM representation and has been primarily used for high-accuracy, non-real-time recognition. Developments have made Sphinx3 “near” real-time, though not yet suitable for critical interactive applications. It is under active development and in conjunction with SphinxTrain provides access to a number of modeling techniques, such as LDA/MLLT, MLLR and VTLN, which improve recognition accuracy.

c. Sphinx4

It is a complete rewrite of Sphinx engine with a goal of providing a more flexible framework for research in speech recognition, written entirely in Java. Current development includes: developing a new (acoustic model) trainer, implementing speaker adaptation (e.g. MLLR), improving configuration management, creating a graph-based UI for graphical system design

d. PocketSphinx

A version of Sphinx that can be used in embedded systems. PocketSphinx can be used with Linux, Windows, MacOS, iPhone and Android.

PocketSphinx for android depends on a library called Sphinxbase that provides common functionality across all CMUSphinx projects which helps it support the following features through various APIs:

- It uses abstract types which allow it to provide stability of source and binary compatibility as well
- It is fully re-entrant, so there is no problem having multiple decoders in the same process
- It has enabled a drastic reduction in code footprint and a modest but significant reduction in memory consumption

The basic functionalities of speech recognition like receiving speech input, converting input to phonemes, building language model and acoustic model, decoding the file stream are provided by PocketSphinx.

How CMUSphinx works

PocketSphinx supports advanced features like word segmentation, obtaining the confidence of the whole word utterance, lattice accessibility and runtime switch between multiple searches. Several search options can be configured by the developer using more than one grammars and language models. It requires to switch between them at runtime for providing interactive environment. The following search modes are provided for PocketSphinx:

- Keyword

It allows the configuration of detection threshold by ignoring the speech other than keywords. It will efficiently search for key phrases only.

- Grammar

It performs the search according to the specification of Java Speech Grammar Format (JSFG) grammar. It doesn’t overlook the words which are not present in the grammar but tries to recognize them.

- Ngram/lm

It uses the language model for recognizing natural speech.

- Allphone

It performs the phoneme recognition using phonetic language model

Switching between searches can be useful in the kind of scenario where the user might want to listen for activation keyword first, once the keyword is recognized switch to ngram search to recognize actual command. Once the command is recognized, user can switch to grammar search to recognize the confirmation and then switch back to keyword listening mode to wait for another command.

There are other speech recognition tools such as Google cloud speech API, AT&T Watson, Microsoft Speech Server and Nuance Recognizer which are however not open source. Proprietary systems offer little control over the recognizer's features, and limited native integrability into other software [23].

3.6 TOOLS FOR IMPROVING ACCURACY

GRAMMARS

A grammar describes a very simple type of the language for command and control. They are usually written by hand or generated automatically within the code. A grammar consists of a structured list of rules that identify words or phrases that the speech recognition engine should attempt to identify in the spoken input [21].

While the built-in language model of a recognizer is intended to represent a comprehensive language domain (such as everyday spoken English), a speech recognition application will often need to process only certain utterances that have particular semantic meaning to that application. Rather than using the general language model, an application uses a grammar that constrains the recognizer to listen to only speech that is meaningful to the application. Grammars usually do not have probabilities for word sequences, but some elements might be weighed

Grammars provide the following benefits:

- Increase in the accuracy of recognition
- A guarantee that all recognition results are meaningful to the application

A speech recognition grammar consists one or more rules. It defines a pattern or sequence of words or phrases (a set of constraints that a speech recognizer can use to perform recognition). If the user's statement matches the pattern, the rule is matched by the application.

Grammars can identify simple one-word commands such as "open" or "print", and more complex sentence structures such as "I'd like to book a flight from Nairobi to Kinshasa, departing from next Tuesday". Grammars typically define a limited vocabulary that is focused on a specific task or set of tasks that a user expects to be able to accomplish by speaking.

A grammar must define structured, logical speech statements that apply to specific situations. It however must be flexible enough to allow slight variations in spoken input to enable a more natural speaking style and to provide a better user experience. For Example, a coffee ordering application must accept and respond to multiple ways that a user can order coffee:

- “I would like a decaf latte”
- “I want a decaf latte”
- “Gimme a decaf latte”

A coffee ordering application must also process and respond to orders for different types of coffee drinks in multiple variations. For example, an application must be able to substitute the word “latte” for “cappuccino” or “mocha” and must be able to insert variations of those drinks such as “decaf” or “iced”. However a coffee ordering application does not need to process and respond to orders for airline tickets; a grammar need only include words and phrases for a specific situation or task.

Rules and semantic definitions can be used to organize a grammars content into logical groupings. This reduces the amount of work that an application needs to perform during recognition by decreasing the amount of information that the application needs to process and by returning only the most actionable information

Purpose of Grammars

A grammar helps an application perform speech recognition in the following ways:

- **Limits vocabulary.** A grammar contains only the exact words or phrases that an application needs to match for successful recognition of spoken input, therefore the recognition engine doesn’t need to search the entire dictionary. Explicitly providing words in a grammar also improves the recognition accuracy.
- **Customizes vocabulary.** A grammar customizes a vocabulary for a particular application. Although a grammar should be flexible and accommodate a number of phrases, grammar structures need to restrict the spoken input to a specific situation or task.
- **Filters recognition results.** A grammar filters the results of recognition that are sent to an application. The recognition engine returns a successful recognition event only if the grammar is matched, otherwise the application would receive many additional recognition results, few of which would have little meaning to the application.

- **Defines semantics.** A grammar can assign an alternate meaning to a word or phrase used to recognize spoken input. For example, assigning the meaning “yes” to any speech inputs “yeah”, “yup”, “yep”, or “affirmative”.

REFERENCES

1. Wikipedia, March 2018. Speech recognition. [https://en.wikipedia.org/wiki/Speech_recognition]
2. Neha Chadha R.C., Gangwar (PhD), Rajeev Bedi, December 2015. *Current Challenges and Application of Speech Recognition Process using Natural Language Processing: A Survey.* International Journal of computer Applications (0975 – 8887) Volume 131 – No.11, [https://www.researchgate.net/publication/291042346_Current_Challenges_and_Application_of_Speech_Recognition_Process_using_Natural_Language_Processing_A_Survey]
3. Kaur Gill, Manjot & Reetkamal, Kaur & Jagdev, Kaur. (2010). Vector Quantization based Speaker Identification. International Journal of Computer Applications. 4. 10.5120/806-1146.
4. Mr. Mahesh Kumar Patil1, Prof. Dr. (Mrs.) L.S. Admuthe2, Mr. Prashant P Zirmite3, Prof. Mr. N.B. Kapase4. A Review on Speech Feature Extraction and Speech Feature Matching Technique. KIET International Journal of Communications & Electronics Volume. No. 2, Issue No. 2, July – Oct 2014, ISSN: 2320 – 8996.
5. Stuart J. Russell, Peter Norvig. Artificial Intelligence. A modern Approach. Prentice Hall, 3rd Edition, 2010.
6. Heather Sobey. Literature Survey – Automatic Speech Recognition. Department of Computer Science, University of Cape Town.
7. Michael Lutter, November 2014. Mel-Frequency Cepstral Coefficients. [http://recognize-speech.com/feature-extraction/mfcc].
8. Wikipedia, February 2018. Acoustic Model. [https://en.wikipedia.org/wiki/Acoustic_model#Speech_audio_characteristics]
9. What is a Language Model? [http://www.voxforge.org/home/docs/faq/faq/what-is-a-language-model]
10. Sharma, Davinder & Atkins, Jamin. (2014). Automatic speech recognition systems: Challenges and recent implementation trends. International Journal of Signal and Imaging Systems Engineering. 7. 220-234. 10.1504/IJSISE.2014.066600.

[https://www.researchgate.net/publication/269931894_Automatic_speech_recognition_systems_Challenges_and_recent_implementation_trends]

11. Kevin Sookocheff, July 2015. Modelling Natural Language with N-Gram Models. [https://sookocheff.com/post/nlp/n-gram-modeling/]
12. Steve Renals Automatic Speech Recognition ASR Lecture 10 January - March 2012
13. Wikipedia, March 2018. Dynamic time warping. [https://en.wikipedia.org/wiki/Dynamic_time_warping]
14. Mohan. B, Jagan & Babu.N, Ramesh. (2014). Speech recognition using MFCC and DTW. 10.1109/ICAEE.2014.6838564. [https://www.researchgate.net/publication/260762671_Speech_recognition_using_MFCC_and_DTW]. Accessed Mar 14, 2018.
15. Jing XinXing, and Shi Xua, 2012. Speech Recognition Based on Efficient DTW Algorithm and Its DSP Implementation. International Workshop on Information and Electronics Engineering (IWIEE).
16. About HTK. [http://htk.eng.cam.ac.uk/]
17. About Julius. [http://julius.osdn.jp/en_index.php]
18. About Simon. [https://simon.kde.org/]
19. Wikipedia, January 2018. CMU Sphinx. [https://en.wikipedia.org/wiki/CMU_Sphinx]
20. Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Luka´s Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motl´ıcek, Yanmin Qian, Petr Schwarz, Jan Silovsky´, Georg Stemmer10, Karel Vesely´. The Kaldi Speech Recognition Toolkit. [http://kaldi-asr.org/doc/about.html]
21. Grammars Overview (Microsoft.Speech), 2018. [https://msdn.microsoft.com/en-us/library/hh378458(v=office.14).aspx]
22. CMUSphinx Tutorial for Developers. [https://cmusphinx.github.io/wiki/tutorial/]

23. Christian Gaida, Patrick Lange, Rico Petrick, Patrick Proba, Ahmed Malatawy, and David Suendermann-Oeft. Comparing Open-Source Speech Recognition Toolkits.
24. X.D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for speech recognition* (University of Eidenburg)
25. Jaime Diaz and Raiza Muniz, *Voice Recognition System,* a journal article, [May 2007]
26. Bhiksha Raj, Rita Singh, *Design and Implementation of Speech Recognition Systems*, a journal article, [2011]
27. Kumbharana, Chanderesh k, 2007, *"Speech Pattern Recognition for Speech to Text Conversion"*, thesis PhD, Saurashtra University
28. Dr. Joseph Picone, *Fundamentals of Speech Recognition: A Short Course,* Institute for Signal and Information Processing Department of Electrical and Computer Engineering Mississippi State University.