Speech recognition for human computer interaction

Ubiquitous computing seminar FS2014
Student report

Niklas Hofmann
ETH Zurich
hofmannn@student.ethz.ch

ABSTRACT
The widespread usage of small mobile devices as well as the trend to the Internet of Things showed that new means of human-computer-interaction are needed. One of many approaches is the usage of voice to recognize the user or given commands. The topic of speech processing has been studied since the 1960's and is very well researched. This report gives an introduction and overview into this broad topic. It presents a model for the speech processing pipeline and shows common techniques for certain stages (namely feature extraction and feature matching). The last part discusses the results of existing systems.

INTRODUCTION
The last two decades saw the development of increasing possibilities where computers or so-called smart devices can be used. This was fuelled on one hand by the rapid shrinking of the used electronics and on the other hand we have an increase in computing and storage capabilities for the same amount of space and energy. This leads, together with better batteries, to the usage of computers in places and situations where it was not possible before. We now find them (in the broadest sense) in almost all aspects of our daily life: from cars to kitchen appliances over to children's toys and mobile phones. But not all of these *smart devices* can make use of the traditional interfaces humans are accustomed to: the keyboard and mouse.
A different approach that has been studied very well, is the analysis of the human voice for means of human computer interaction. Even though the process of speech processing, and more precise speech recognition, has been around since the 1960s, only recently have mobile devices had enough processing power and storage to be able to perform speech processing. This topic can be divided into three different applications:

- speaker recognition
- speech recognition
- language recognition

In the following sections an overview of the two topics speaker and speech recognition will be given. The application of speech processing to language recognition will not be treated in this report.

Speaker recognition
The goal of a speaker recognition system is to determine, to whom a recorded voice sample belongs. To achieve this, possible users of this system need first to enroll their voice. This is used to calculate a so-called model of the user. This model is then later again used to perform the matching, when the system needs to decide the owner of a recorded voice section. Such a system can either be built text-dependent, meaning that the user needs to speak the same phrase during enrollment and usage of the system. This can be seen similar to the setting a password for a computer, which is then required to gain access. A recognition system can also operate text-independent, so that the user may speak a different phrase during enrollment and usage. This is more challenging to perform but provides a more flexible way the system can be used. Speaker recognition is mainly used for two tasks:

Speaker verification assumes that a user claims to be of a certain identity. The system is then used to verify or refute this claim. A possible use case is to control access for a building. Here, the identity claim could be provided by the usage of a secondary system like smart cards or fingerprints.

Speaker identification does not assume a prior identity claim and tries to determine to whom a certain voice belongs. These systems can either be built for a closed group, where all possible users are known to the system beforehand. Or it can be used for a open group, meaning that not all possible users of the system are already enrolled. Most of the time this is achieved by building a model for a “unknown speaker”. If the recorded voice sample fits this “unknown model” the speaker is not yet enrolled. The system can then either just ignore the voice or build a new model for this speaker, so that he later can be identified.

Speech recognition
The idea behind speech recognition is to provide a means to transcribe spoken phrases into written text. Such a system has many versatile capabilities. From controlling home appliances as well as light and heating in a home automation system, where only certain commands and keywords need to be recognized, to full speech transcription for note keeping or dictation. There exist many approaches to achieve this goal. The most simple technique is to build a model for every word that needs to be recognized. As shown in the section pattern matching this is not feasible for bigger vocabularies like a whole language. Apart from constraining the accepted phrases, it must be considered if the system is only used by a single individual, so-called speaker dependent, or if it should perform equally well for a broad spectrum of people, called speaker independent.

SPEECH PROCESSING PIPELINE

Although speaker recognition and speech recognition systems achieve different goals, both are built from the same general structure. This structure can be divided into following stages:

- *Signal generation* models the process of speaking in the human body.
- *Signal capturing & preconditioning* deals with the digitalisation of voice. This stage can also apply certain filters and techniques to reduce noise or echo.
- *Feature extraction* takes the captured signal and extracts only the information that is of interest to the system.
- *Pattern matching* then tries to determine to what word or phrase the extracted featured belong and concludes on the system output.

Signal generation

The production of voice inside the human body is a complicated process that involves many structures like the lungs, the vocal tract, vocal folds (also known as vocal cords), the oral and nasal cavity. A simplified model of the human body is shown in Figure 1. The lungs provide a stream of air, here modeled as the power supply. This stream is then modulated via the vocal folds into, in this model, three different wave forms:

- *Periodic puffs* are responsible for periodic sounds as in */a/* or */o/*, the so-called *voiced sounds*. This is generated by the closing of the vocal folds, which then leads to an increase in pressure resulting in a forced opening of the folds and thus again to a decrease in pressure. The vocal folds can close again and the cycle is repeated periodically.
- *Noise* occurs when the air stream is directed over halve closed vocal folds. This generates the *fricative sounds* as in */f/* or */s/*.
- *An impulse* is generated similarly to the periodic puffs, but will only involve a single period of the cycle. This leads to the *plosive sounds /p/*, */t/*, etc.

This source signal will then propagate through the vocal tract which consists of the throat, the nasal and the oral cavity. This tract will act as a sort of filter that changes and distorts the source, seen in figure 2. This happens mainly through resonance and reflections and depends on the shape of the vocal tract. During speaking this shape will change continuously which thus changes the behaviour of the filter. Although this happens continuously, it is appropriate to assume that the vocal tract remains stable when voicing a sound and only changes on transitions from one sound to another. This assumption eases the workload when trying to estimate the vocal-tract filter from a recorded voice sample as further described in the feature extraction section.

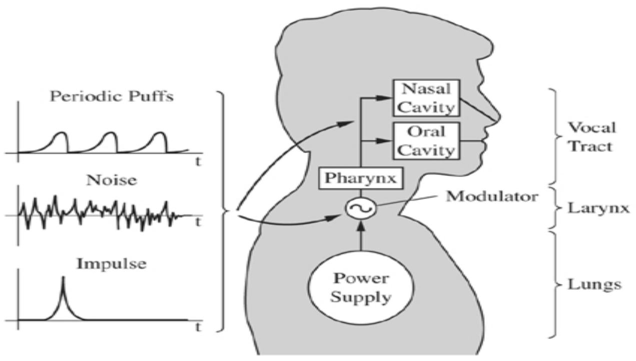

Figure 1: Voice generation in human body from [11]

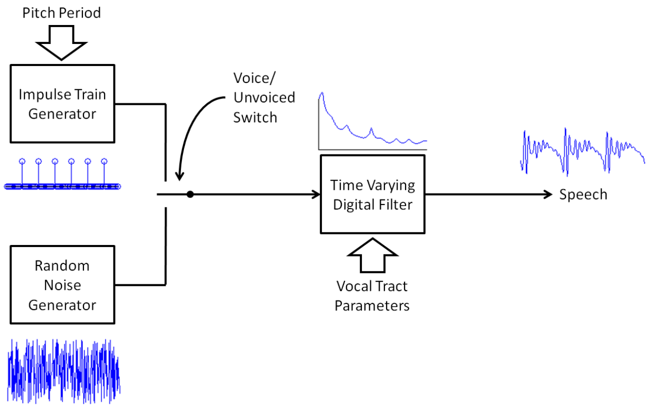

Figure 2: Simplified voice model from [10]

Signal capturing & preconditioning

This stage performs multiple tasks. The first is to record and digitize the voice. This is done with a microphone and certain technical specifications have to be kept in mind, like the bandwidth. When digitizing one has to take care not to introduce artefacts through aliasing, done by choosing the sample frequency to be at least twice the highest frequency desired to be captured (Nyquist-Shannon sampling theorem). Further steps can include the reduction of noise and echo, which is not further treated in this report. Once a signal is acquired start and endpoint detection is performed, which can be done for example by thresholding. The start and endpoint are later used to limit the analysis of the signal onto the parts which really contain speech.

Feature Extraction

Given a signal that has been reduced to only the voiced part, this stage now extracts the information relevant for the matching. As mentioned in signal generation the vocal tract (and thus the voice itself) can be assumed to be stable for a short period of time. This time ranges from 20 to 40 milliseconds. The first step is then to divide the signal into small frames of duration of around 30 milliseconds. To not lose information at the borders, the frames normally overlap each other by 10 to 20 milliseconds. A window function is then applied, as practically every technique to extract features depends on the analysis of the spectrum. Each frame will then lead to a series of features that are extracted. This is normally called a feature vector. A whole signal will thus lead to a series of feature vectors that can then be classified in the matching stage. There are two prevalent techniques to extract features from a frame:

Linear Prediction assumes that the vocal tract performs like a linear filter that uses the current source u_n as well as previous outputs S_{n-k} to model the current output S_n :

$$S_n = -\sum_{k=1}^{p} a_k \cdot S_{n-k} + G \cdot u_n$$

Because the source signal u_n is generally unknown, a modified formula is used:

$$\widehat{S_n} = \sum_{k=1}^{p} a_k \cdot S_{n-k}$$

The parameters a_k are then calculated by minimizing

$$e_n = S_n - \widehat{S_n}$$

The so-called prediction order p determines how far back the filter will incorporate past output signals and is determined among others by the sampling frequency used to record the signal [2]. When only linear prediction is used for feature extraction, the feature vector for one frame then consists only of the prediction coefficients a_k.

Cepstral Coefficients has its origins in the application of the convolution theorem. The response S in time domain of a filter can be calculated by the convolution of the source U with the frequency response H of the filter. If the fourier transform $F\{\cdots\}$ is then applied to the output of the filter, this simplifies to a multiplication in the time domain (here $*$ is used to signify convolution):

$$S = U * H$$

$$F\{S\} = F\{U\} \cdot F\{H\}$$

To further ease the separation of the filter from the source the so-called cepstrum can then be calculated by taking the inverse fourier transform of the logarithm of the spectrum:

$$F^{-1}\{log(|F\{S\}|)\} = F^{-1}\{log(|F\{U\} \cdot F\{H\}|)\}$$

which further simplifies to:

$$F^{-1}\{log(|F\{U\}|) + log(|F\{H\}|)\}$$

$$= F^{-1}\{log(|F\{U\}|)\} + F^{-1}\{log(|F\{H\}|)\}$$

As these are discrete signals, the resulting cepstrum will also be discrete and the different values are called the cepstral coefficients. It can be shown that the source signal can be described by the higher coefficients whereas the filter is reflected in the lower coefficients [1]. This allows for the filter to be extracted by the simple means of a low-pass filter applied to the cepstrum. Again the cutoff coefficient of the low-pass filter is a parameter that has to be adjusted to the whole system. At the end the feature vector consists of the chosen coefficients.

Matching

Once a voice sample has been converted to a series of feature vectors it can be fed into the model to determine who or what has been spoken. This is done by building a model using the extracted features. In this report two types of models will be presented:

- *Template matching* that uses prerecorded samples as a sort of stencil
- *Statistical methods* that use a bigger set of prerecorded samples, also-called a training set or training data, and apply methods from machine learning

When a model has been built, a sample can then be compared against the model and some output is generated. In case of classical pattern matching methods this will result in a distance measure. If the model is built with means of statistical approaches, then the output will be a probability, that the sample has been generated by this particular model. Here the signal to classify is called the sample and, depending on the method used, it is compared to either a template or a model.

Pattern matching is the most simple approach and makes very strict assumptions:

- The sample and the template are of equal length, thus consist of an equal number N of feature vectors each.
- Sample and template are spoken with equal speed.

This method works by assuming that each feature vector of the sample S or the template T can be seen as a point in a higher dimensional space. The resulting measure of how close both signals are is calculated by taking the pairwise distance of two corresponding feature vectors which is then summed up over all vectors of the sample.

$$d(S,T) = \sum_{i=1}^{N} |S_i - T_i|$$

It is clear that this method is very sensitive to different speech patterns (length, pronunciation) and thus is not used anymore.

Dynamic time warping is an approach that fixes the shortcoming of the simple pattern matching. The idea is to stretch and shrink the sample or template to account for the different speed and length. This is done by the technique of dy-

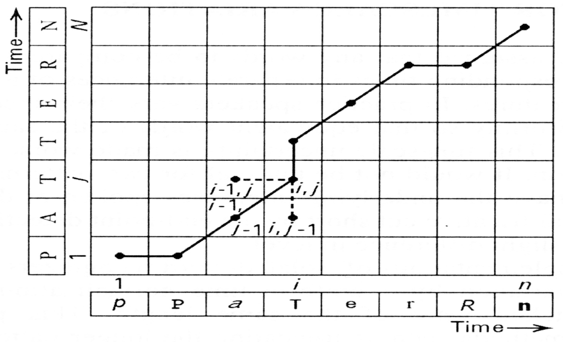

Figure 3: Dynamic time warping (DTW) from [1]

namic programming. As seen in figure 3, this method finds a timeline through the sample and template to minimize the accumulated distance along that path [1]. In the figure a horizontal step corresponds to the passing of time in the template whereas the vertical axis represents the time in the sample. The more a chosen path deviates from the diagonal, the more the timescale has had to be stretched or shrunk. The parameter for this method lies in the penalization of paths that deviate too much from the diagonal, as this shows that sample and template are not similar. Once the best warped time path has been found, the method then calculates again the distance of the two signals similarly as in the simple pattern matching. The benefit of this method lies in its simplicity and that it does not need an extensive training set. But it is only feasible to recognize phrases for which a template has been prerecorded. This makes it appropriate to identify keywords and commands.

Hidden Markov model (HMM) is a method from machine learning that has been applied to speech processing since the 1980s. Used for speech recognition it accounts for the fact that template matching methods use all feature vectors the same, even though not every feature vector needs to be of the same importance to recognize a certain word of phrase. This "information content" for a certain feature can even differ from word to word and is in general not uniform. To alleviate this shortcoming, statistical methods were developed, where multiple utterances for the same word will be consolidated into a single model for that word. The resulting model should then be able to recognize different pronunciations and speaking speeds.
A hidden Markov model tries to capture a process, where the underlying mechanism transitions through some hidden states which are not directly observable. Each of these states in turn will have a certain probability to emit a feature that can be observed. For example the tossing of two different, unfair coins could be modeled by an HMM consisting of two states, see figure 4. Here the coefficients $a_{i,j}$ are the probabilities of changing state after a coin has been flipped. Each state has its own probability density function P_k which models the probability of getting a *head* or *tail* (e.g. $P_1(H) = 0.9$ for a 90% chance of flipping a *head* in State 1).

A good analogy for hidden Markov models applied to speech recognition is that the states correspond to the shape of the vocal tract and that the observable features are the voice signal for a single frame. This approach could then be further adapted to build an HMM for all words one wants to detect. The problem with this strategy lies in the fact that the model for longer words will get very complicated and only the words already in the training set will be recognizable. A better approach is to build an HMM for every sound, so-called phoneme, that exists for a certain language (ca. 45 in American english). A corresponding model would then consist of 3 states (e.g. sound */a/*

- 1st state for transition: */sil/* $\rightarrow$ */a/*
- 2nd state for the sound itself
- 3rd state for the transition: */a/* $\rightarrow$ */sil/*

This solves the problem for recognizing words outside the training set but suffers from a different problem. The sounds in the human language are not context-free. This means that in a triplet of sounds (e.g. /cat/) the leading and the following sound can alter the pronunciation significantly. This lead to the usage of so the so-called *triphone* model, which generates an HMM for each triphone in the training set or even for each triphone in the language. This approach comes with a big computational cost, as for the English language there exists around $90'000$ different triphones.

Regardless of the chosen system (single sound or triphone), the built models can then be chained together to recognize words that were not originally in the training data. This marks a big advantage over the template matching methods. See [1], [3], [4] and [6] for more detailed information about hidden Markov models.

Comparison DTW against HMM

To give an idea how the presented methods compare against each other, the results of [8] are here briefly presented. The experiment compared the two feature extraction methods linear prediction and ceptral analysis together with the two matching methods dynamic time warping and hidden Markov models. The comparison was performed on a set of eight female and eight male speakers, each speaking the digits zero to nine. The training data consisted of 500 words, each digits spoken 50 times. The models were then evaluated on a test set containing 100 words with each digits represented 5 times. As one can see from figure 5 the statistical approach outperforms the template matching with around 10% to 15% more accuracy for the same feature extraction method used. It can be further deduced, that the usage of cepstral analysis

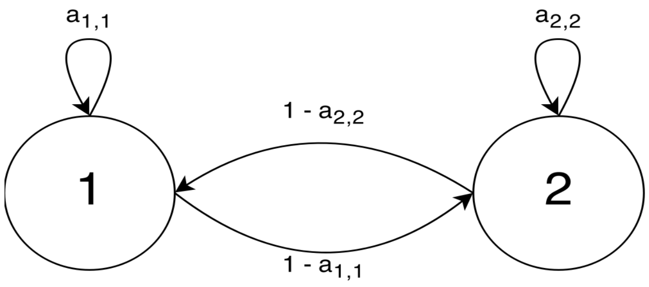

Figure 4: Simple hidden Markov model from [3]

Types of features	Frame length	Recognition Accuracy in % using DTW	Recognitio n Accuracy in % using HMM
LPC	13	69	86
LPC+Δ+Δ^2	39	76	91
MFCC	13	77	90
MFCC+Δ+Δ^2	39	86	94

Figure 5: Comparison DTW vs. HMM from [8]

in comparison to linear prediction also increases the accuracy by 10% to 15%.

APPLICATION TO MOBILE DEVICES

When applying the process of speech or speaker recognition to mobile devices multiple issues arise:

- limited power supply
- limited storage
- limited computing power

Because of this factors the system cannot be too complex and must not use too much battery. Further consideration must be made in terms of speed of computations. A system that takes significantly longer than the voiced part to perform its calculations is not desirable. Neither is a system that runs in real time but provides no accuracy.

Local system

A possible solution, as described in [7] to these problems is to use two models to recognize the owner of a mobile device against the background noise and other speakers. This model only detects a certain keyword. Upon successfully perceiving the keyword from the correct speaker, the system activates further capabilities to recognize commands. This approach minimizes the computational effort when the detected voice is not from the owner of the device as only two models need to be considered. Experiments done on a ultra mobile personal computer with a CPU of 500 MHz, see [7] show, figure 7, that the runtime of a speech recognition system can vary greatly depending on the method used for the matching (template based against HMM). This demonstrates well that although hidden Markov models are more precise, they come with a big increase in computational cost. Another disadvantage of HMM is that they rely on the availability of a big training set to calculate statistics on. Figure 6 shows that the equal error rate (the system classifies an equal amount of false positives and false negatives) of a hidden Markov model can be even worse, provided not enough data can be used. Thus the decision what system to use does not only rely on the accuracy alone but also has to consider factor like runtime and need for training data.

Cloud based system

A new approach to speech processing on mobile was possible with the widespread deployment of reliable internet access

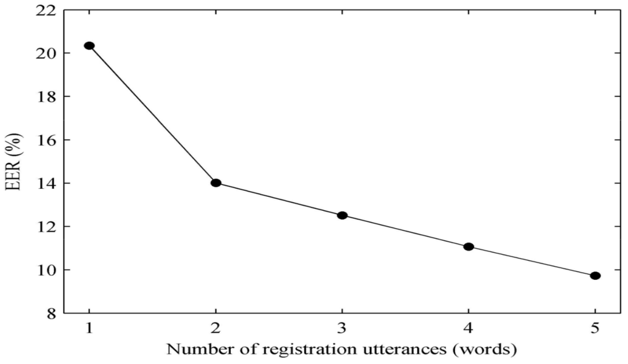

Figure 6: Dependence of hidden Markov models on training data from [7]

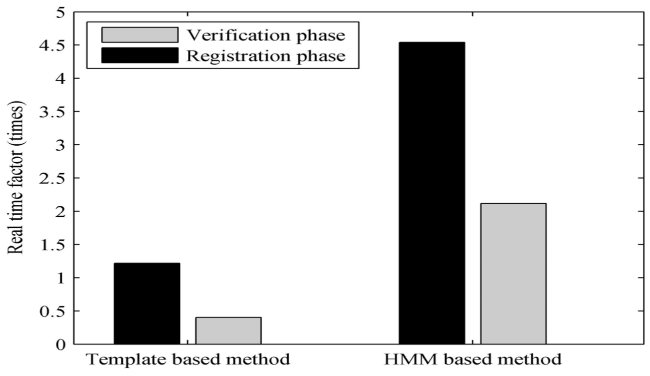

Figure 7: Runtime for training and verification on mobile device from [7]

on mobile devices. This enabled the local capturing of the voice signal, which then is transferred over the network to a remote server where the whole feature extraction on matching process will be done. At the end, only the output of the system has to be transferred back to the device. This circumvents the posed limitations on mobile devices but has the drawback of relying on internet access. But a huge gain is achievable by using modern data centers that offer orders of magnitude more computing and storage capabilities compared to a modern smart phone. Preliminary experiments done by Google [9] show that it is possible to use bigger feature vectors (39 dimensions, both linear prediction and cepstral coefficients) together with the application triphones to hidden Markov models. It is notable that these systems make heavy use of a so-called jem language model to restrict the search space when performing the recognition on a voice sample. In this particular example the language model could be trained on typed search queries (around 230 billion words). It is shown that the out-of-vocabulary rate (the percentage of words not modeled in the language system) can be halved when local accents are considered [9]. With the increase in computing capabilities of mobile devices, there are already systems that use the cloud when internet access is available, but rely on a reduced set of features when working only locally. This trend blurs the line between fully local and cloud based systems.

CONCLUSION

This report gave an introduction into the broad topic of speaker and speech recognition. It showed the coarse structure of a speech recognition system and gave examples for common practices for both feature extraction and matching. It showed the differences in the presented techniques in both accuracy as well as computation required. The adaption to mobile devices has been discussed briefly with a short section about modern cloud based systems. As this subject has been researched for about 50 years, it is out of the scope of this report to give a comprehensive view on the topic. This report sets the focus rather on established methods and concepts than trying to incorporate the latest technologies, as this topic is still of great interest for scientists and engineers.

REFERENCES

1. Holmes, J., Holmes W. *Speech Synthesis And Recognition 2nd Edition.* Taylor & Francis, New York NY, USA 2001

2. Cinneide, A., Linear Prediction: The Technique, Its Solution and Application to Speech. Dublin Institute of Technology, 2008

3. Rabiner, L., A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE, vol. 77, no. 2* pp. 257-286, 1989

4. Campbell, J.P. Jr., Speaker recognition: a tutorial. In *Proceedings of the IEEE, vol. 85, no. 9* pp. 1437-1462, 1997

5. Bimbo, F., et al., A tutorial on text-independent speaker verification. In *EURASIP Journal on Applied Signal Processing* pp. 430-451, 2004

6. Wilpon, J.G, Rabiner, L., Chin-Hui, L., Goldman, E.R, Automatic recognition of keywords in unconstrained speech using hidden Markov models. In *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 38, no. 11* pp. 1870-1878, 1990

7. Hyeopwoo, L., Sukmoon, C., Dongsuk, Y., Yonkserk, K. A Voice Trigger System using Keyword and Speaker Recognition for Mobile Devices. In *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 38, no. 11* pp. 1870-1878, 1990

8. Sharada, C., Vijaya, C. Comparison of DTW and HMM for Isolated Word Recognition. Proceedings of the Internation Conference on *Pattern Recognition, Informatics and Medical Engineering*

9. Schalkwyk, J., et al., Google Search by Voice: A case study. In *Advances in Speech Recognition.* Springer, 2010

10. Gutierrez-Osuna, R. CSCE 689-604: Special Topics in Speech Processing, Texas A&M University, 2011, courses.cs.tamu.edu/rgutier/csce689_s11

11. Quatieri, T. F., *Discrete-Time Speech Signal Processing: Principles and Practice.* Prentice Hall, 2001