Automatic Speech Recognition

Some SIRIous business

Ana Crnkovic, Karlo Loci,
Katarina Petrović, Mario Zelić

***Abstract*— This project shows how to make a simple speech recognition program in one week time. Firstly, speech is separated from noise. Afterwards, the word recording is matched to an HMM of a word it most likely represents. These HMMs are learned from data recordings. The results depends on the features we use. When using features depends only on amplitude and frequencies, the results are poor. We also use professional features which give us good results. We change our data set division by leaving out recordings of one team member and then we would use them for testing. Interestingly, the results were better with lower pitched test voices.**

I. Introduction

A. Problem description and motivation

Since its beginning, speech recognition software has been a not-so-rare topic in scientific magazines. The basic idea behind speech recognition is analyzing a recorded sound signal, detecting where the speech is in the signal and recognizing a word out of that signal. Although the technology and software behind it are already pretty advanced, their implementation in improving the efficiency of doing daily tasks is limited regarding its capabilities. Today, there are many speech recognition apps such as “SIRI” on IOS, “Skyvi” on Android and “Cortana” on Windows Phone.

B. Project Goals

That is where our project comes in. Our goal is to create simple yet useful speech recognition software that will use the sound signals, analyze them and create text out of them. We use HMMs (Hidden Markov Models) to calculate which word is being said. Hidden Markov Model represents probabilities of a letter repeating itself or advancing to the next letter. It can be used to calculate how likely it is that a word is said with a certain speech signal. We compare the sound signal to HMMs that are trained on other sound signals previously recorded. The word of the most likely HMM is displayed.

II. Background

A. Sound

A sound is a vibration of particles trough some medium. It is characterized by amplitudes and frequencies. Amplitude is the difference between extreme values (peaks and valleys) while frequency is the number of occurrences of a repeating event per unit time.

B. Speech

Humans exhale air while speaking. Air passes between two vocal folds which sets them into vibration, making a sound which passes through our vocal tract to the mouth. Going through our vocal tract, some frequencies are filtered based on the shape of our vocal tract making the final sound a superposition of waves of those frequencies. Therefore any speech, because of the way sounds are summed, is a periodic wave in the Amplitude – Time graph.

C. Hidden MarkovModels

A Markov model is a model in which each state depends on the state before that. Figure 1 gives an example of a Markov model.

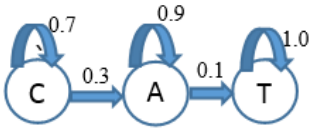


Figure 1 Markov Model

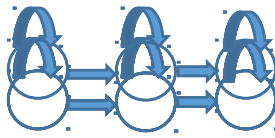
In the recording of the word “c c c c a a a a a t t t”, it is easy to know which word it means since we know the phonemes said in each window and they are obviously correct, that is they make a word with meaning.

In the application of speech recognition, one is not sure of which phoneme is pronounced in each state. The computer could get the following sequence of windows: “C c g c a a a a t t t” which no longer satisfies the Markov model. A human would recognize the mistake due to the knowledge of the language and replace “c g c” with the more likely “c c c”.

A Hidden Markov Model solves this problem. The phonemes in our model are then unknown or hidden. This is why this kind of model is also called the Hidden Markov model (or also known as HMM).

Given a sequence of windows one can calculate how likely it is that the model will produce the sequence. One can calculate the transition probabilities and the Gaussian parameters from examples (recordings).

More details one can find in Rabiner’s tutorial [1].



III. METHODS

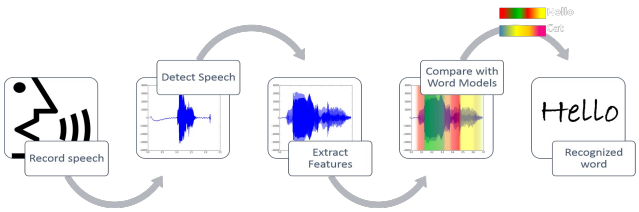


Figure 2 Automatic Speech Recognition

A. Speech Detection

First we have recorded our signal using a microphone and computer. After plotting it on a graph, we ended up with a function, looking like the Figure 3.

Signal was plotted on a plot amplitude of a wave signal over a time-domain. After plotting function it is needed to smooth it. The function is smoothed using windowing. Windowing a function basically means dividing a function into a final number of small pieces which can overlap. The sound signal is windowed and the squares of average amplitude of each window are calculated. After plotting that on a graph amplitude over a time you get something similar to the Figure 4. When we got that function, we chose the threshold by comparing the speech and background parts and calculating some average value between the amplitudes. That is how we distinguished between the speech and silence (background noise).

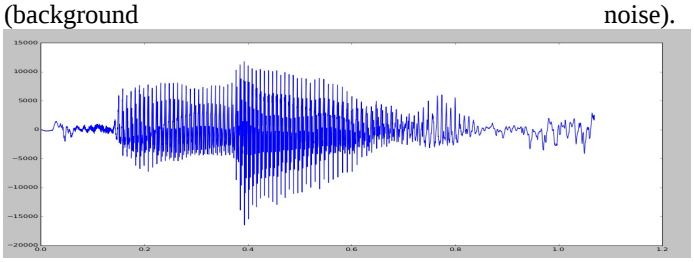


Figure 3 Original speech signal

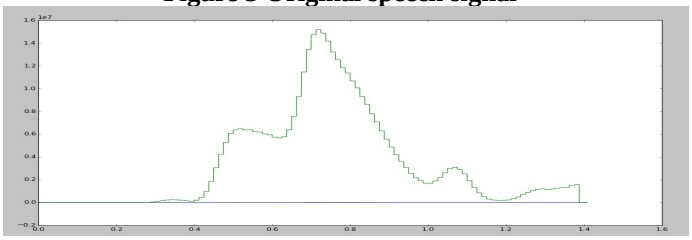


Figure 4 Smoothed speech signal

B. Features

Before actual feature extraction, we need to preprocess the window. The first type of preprocessing is using a Hamming window. If you use a Hamming window you basically switch the rectangle shape to the shape showed on a Figure 5. The reason you do this is because then you avoid adding unnecessary frequencies to your sound signal, therefore you get a better quality.

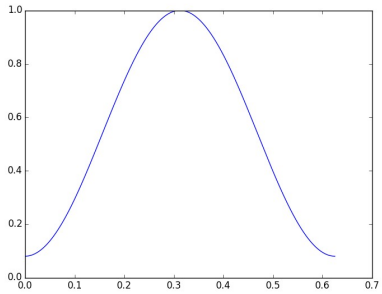


Figure 5 Hamming window

The second preprocessing is called pre-emphasis. Pre−emphasis (Figure 6) is a type of a filter inside a frequency domain used to increase the amplitude of higher frequencies and decrease the amplitudes of lower ones. We use pre−emphasis to more or less equalize the signal inside a frequency domain, also our brain does the same kind of filtering to equalize the frequencies of our most sensitive hearing range (1000−5000 Hz) to make the signal understandable.

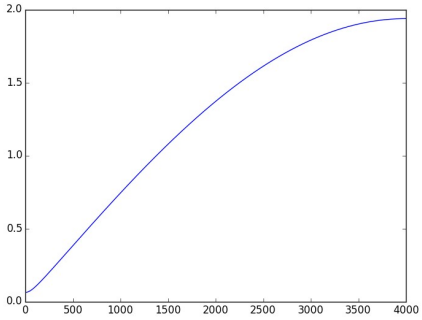


Figure 6 Pre-emphasis filter

Features represent a specific characteristic of a window which can easily distinguish it between others and make it a significant one. In this part of a method, the best feature is found and used for future measurements.

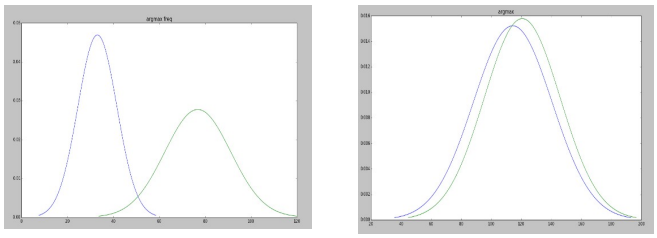


Figure 7 Left: A good feature. Right: a bad feature.

On Figures 7 there are examples of a good feature and a bad one, respectively given. The feature is better if an overlap area of the two Gaussian functions (vowels, for examples E and A) is smaller. It is concluded that the best feature was maximum frequency of a window. The features which have been tried as well are maximum amplitude, average amplitude, standard deviation, minimal amplitude

C. Word Recognition

Here in this part, Hidden Markov Models were made for each word from a database. We calculated the probabilities for

each path in model (we learned our HMM). After calculating the probabilities, the focus was on finding the probability that from a specific HMM model you will be able to get a particular word which has previously been recorded. After calculating this for each HMM, the maximum probability was found and that is basically our word. These are some methods which are used in this project:

1) The method for learning a HMM. In this method, program takes words from a database and first sets up a transition matrix. The transition matrix, in the beginning, can only give you information about the connectivity in a graph (a HMM is a directed graph), when there is a path between the two vertexes, the value of a matrix field is 0.5, only on a field transition_matrix (n-1,n-1), the value is 1.0. After that it'll calculate the probabilities for the next letter (vertex) and transition matrix will change.

2) Method for making a feature matrix using our best feature (maximum frequency), and some professional features such as Linear Predictive Coding and Cepstrum coefficients.

3) Method for classifying a word, aka finding the best HMM model (calculates the probabilities that an hmm model suits our word)

IV. Results

In order to test the software, we used a database with ten words. For each word, the database contains several different recordings from different people. The data is divided in training data and testing data.

A. Training and Testing Data

Training data is used so the software can learn the HMMs for every word in the database. Testing data is used for software testing so that we can evaluate the software and determine if it should be changed or improved. They cannot contain the same recordings.

B. Experiments

The baseline (the minimal percentage of correct answers needed so that the software would do better than random guessing) is determined by the probability of one picking the right word from the whole database. Our database had 10 words, so the baseline is 10%.

Table 1 Feature experiments

Test results	Our features	LPC	Cepstrum	LPC +Cepstrum
Preprocessing	31.33%	76.00%	82.67%	86.00%
No preprocessing	41.33%	54.67%	21.33%	-----

If you look at the table, you can see that the combination of the LPC and Cepstrum with preprocessing gives the biggest percentage of correct answers because they are oriented towards the frequency domain, and our features are oriented towards both time and frequency domain.

Table 2 Multiple Speakers

Leave one out	Percentage of correct answers
Karlo	54.54%
Mario	53.89%
Katarina	31.33%
Ana	29.89%

In a second experiment we test with putting in the training data of three people from our team, while the testing data was from the fourth man/woman.

V. Discussion

With the project, we have confirmed that speech recognition is a very useful software which has many advantages and much more applications in the world.

A. Good Features

Finding features of windows is the most important part of the speech recognition software (beside HMMs) because it gives back a broad spectrum of data which is different for all the windows, but the main look of them is similar. That is why it is important to use good features since they make sure that the software extracts precise data which can be used later for the HMMs.

The first time the software was tested, the features which were used were the ones we came up with and we did not use preprocessing of the sound so we got the results which are much higher than the baseline. Implemented LPC and preprocessed windows return much better results. This is because of the nature of the LPC which is based on the vocal tract and the frequency of voice.

B. Preprocessing

Preprocessing is a procedure which cleans the frequency domain by applying pre-emphasis and the Hamming tool which were discussed earlier in the report. Preprocessing the window is important before you want to extract features like LPC and Cepstrum since they are based only on the frequency domain. By preprocessing, the time domain is altered so the features we came up with score lower, near the baseline. Without preprocessing the features, which we came up with, actually give better results because the features are in frequency domain as well as in the time domain. LPC and Cepstrum do not score better than in the preprocessed windows.

C. Multiple Speakers

The results of "leave one out" vary from 29.9% to 54.54%. Results which women in our team got, were lower than the results from the men. We believe that is because of the different frequencies of male and female voices.

VI. Conclusion

We have developed software that records sound, detect speech from it and recognizes what word it is based on

its features such as maximum amplitude, average amplitude etc.

We recorded several words, made a database of HMMs out of them and compared the word that is being said to the database. Then we compared the words we recognized to the actual words being said and got our percentages of correct guesses from them.

From our results you can see that male voices get higher percentage of correct guesses than female voices. We do not know why this is, but we think that the pitch of the voice and quality of microphones we used play a role in it.

All in all, this software can be used to make our communication through electronic devices easier by replacing visual communication (textual) with our most natural one, vocal.

REFERENCES

[1] Rabiner, Lawrence R. "A tutorial on hidden Markov models and selected applications in speech recognition." Proceedings of the IEEE 77.2 (1989): 257-286.