Differentiating words in speech

Saragadam R V Vishwanath
EE10B035
Department of Electrical Engineering
IITMadras.

November 24, 2011

Abstract

Speech recognition is one of the major challenges in the field of artificial intelligence. The human brain can handle speech so easily that, speech process is considered a very trivial task. However, speech recognition is a huge research field. The following document will present a brief description of the process of speech recognition, the methods used and the reliability of the process.

Further, the document will also present the working of an open source speech recognition engine, called *julian*. The software will also be compared with the inbuilt speech recognition software in $Microsoft^R$ $Windows^R 7$.

Introduction

Humans learn speech from their childhood itself, without any specific instruction, but by mere observations and usage. Speech is so fundamental a trait that it is forgotten that it is a very complex process indeed.

The article is about the structure of speech, the rules it follows and the porting of the mechanism into computers for the process called speech recognition. The article mostly focuses on the technology involved in detection of speech and its limitations.

Speech recognition is not so easy job, not only due to the complex nature of the speech structure, but due to the enivironmental and the other external conditions. For example, noise happens to be one of the major wrong doers in detection and processing of speech. The article will present a case, where noise changes the output of the speech recognition software.

One more contributing factor is the human mind condition. Humans have different mindset at different times, and this leads to different types of generation of the same sound. Computers have to be enough flexible to allow this changes while processing the speech. The program *julius* has been used to study these subtle and not so subtle changes and the outputs are tabulated. The document will also gave a very brief introduction to the *Hidden Markov Model* method of speech detection.

Literature Review

Man Vs. Machine

Unlike computers, which rely on ultra fast linear instruction execution system, the human brain works on a completely different paradigm. The human brain works on what is known as the massive parallel processing. Instead of processing huge complex processes at a high speed, the human brain processes very simple instructions at a very low speed but at a very large scale(The neurons here are the processors). This type of instruction computing is called *Artificial Neural Network*. The speech recognition system exploits the power of the artificial neural network computing to extract the data.

Two concepts will be introduced in very brief to help further understanding

Markov modelling

Any process which obeys markov property in stochastic process is called a markov model. In stochastics(A study of random processes), *markov* property says that the conditional probability distribution of future states only depends on the present state, not on the series of the past states that have occured. Any further description of the modelling will render the paper completey mathematical.

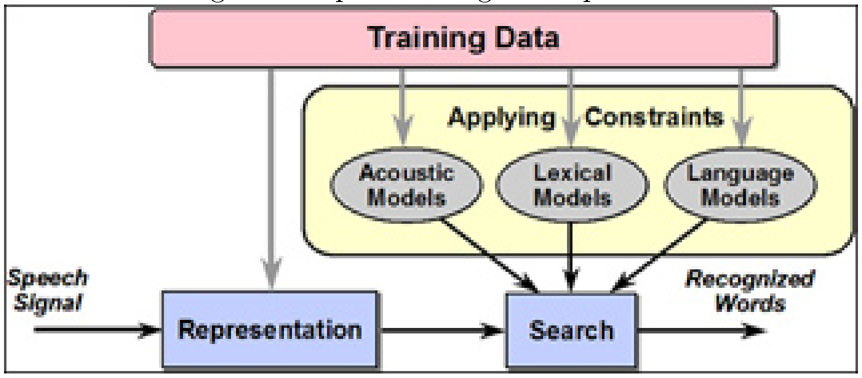
Hidden Markov modelling

A *markov* model in which the intermittant states are not completely visible is known as *hidden markov modelling*. In such an approach, observations related to the state are available but not sufficient to determine the state. This process is the most recent backend in the speech detection process.

Speech recognition

Speech recognition is a multistep process. Initially, the speech has to be confirmed as speech in noise or noise in speech. Fortunately, for speech recognition in most of the cases, speech dominates the noise. But that does not better the situation.Once the speech is 'cleaned', it is processed. This processing of speech consists of two separate modelling.

Figure 1: Speech recognition process



- The first one is the acoustic modeling. The acoustic modeling is the library of sounds, as spoken by the humans. The modeling will take care of the pronounciation, breaks, the characteristic voice pattern etc. More technically speakin, it takes care of the phonemes and the allophones generation.
  Clearly, the acoustic modeling requires the human voice samples as output.

- The second one is the language modeling.It takes care of the word formation, the sentence patterns, possible word combinations. Technically speaking, it takes care of the phonetics of the speech. It requires programming the vocabulary and the grammar.

Both combined will give the speech to text conversion program.

Windows 7 Recognition system

$Microsoft^R Windows$ has an inbuilt speech recognition system, for assitance in operation of the system.It is close source program and not flexible. It does its job good upto a certain extent. If the user has a typical american accent, the speech recognition works very good. However, if the voice is deviant, the recognition goes erroneous. The program will not be probed into further due to license issues.

Julius

Julius is a free, open source, large vocabulary speech engine by $voxforge$.Julius was initially made for the recognition of japanese speech but was later ported to english also. Julius is very flexible, the reliability depends on how well we 'train' the engine. By training, the following process is meant

- The acoustic model has to be intially built to help the engine connect the voice of the user and the text to be output.This is a tedious job, but completely eliminates the constrain of accent of language, since, a person speaking indian english can traind the engine accordingly to recongnise indian english accent.

- The language model is written in two different files, the *.grammar* file and the *.voca* file.The grammar file has all the possible definitions and the combinations for the speech fragments. The vocabulary file has the phonetic definition of the words.This work is very generic and is independent of the acoustic model. However, after compiling, the corresponding acoustic model will be searched and if not found, will raise and error.

This ability of julius to be able to be trained is the most powerful tool. It can virtually encapsulate all the language models, provided the user is ready to train the engine.

The further focus will only be on julius, is examples and the usage of the engine for adding vocabulary.

Data

The *julius* speech engine has been installed in *ubuntu* for the purpose of testing. The julius speech engine will be critically analysed with 4 words, the digits, *seven* and *eight* and the names *steve* and *maclean*.The words are chosen, since they are a part of the sample dictionary installed with julius.

Usage of julius

The usage of julius is somewhat non-intuitive. Firstly, it is installed in ubuntu using the following command

```
$ apt-get install julius julius-voxforge
```

julius is the speech engine and julius-voxforge is the acoustic and grammar modelling program for english language(still under development).

The sample file, containing the *sample.grammar* and *sample.voca* file were compiled using the following command

```
$ mkdfa sample
```

mkdfa is the command used in the linux terminal for compiling the grammar and the vocabulary. Once done, the *julian.jconf* file, which has the execution rules is utilised along with the sample grammar with the following command

```
$ julius -input mic -C julian.jconf
```

This step readies the terminal for input via the computer mic. Now, the text corresponding to the speech will be dynamically output to the terminal.

Initial trials

The initial path was a difficult one. The inbuilt mic in the laptop did not work,even after much searching and meddling around with the computer. Finally a workaround, which turned out to be a boon was found. The sound was created separately using the opensource sound editing software *Audacity*. This would generate a *.wav* file which can be used as input to the speech engine.

The followin command helped to use *wav* file as input to the engine

```
$ julius -input file -C julian.jconf
```

Once this worked, various voices were given as input and analysed.

Now, an attempt was made to write and ingenious grammar, with the help of the voxforge website tutorial. The following grammar and vocabulary was written.

Listing 1: Grammar file

```
S  :  NS_B HMM SENT NS_E
S  :  NS_B SENT NS_E

SENT:  TAKE OBJECT PLEASE
SENT:  TAKE OBJECT
SENT:  OBJECT PLEASE
```

```
SENT: OBJECT
SENT: PLEASE
SENT: CALL NAME PLEASE
SENT: CALL NAME
SENT: NAME
```

Listing 2: Vocabulary file

```
% NS_B
<s>                 sil

% NS_E
</s>                sil

% HMM
FILLER              f  m
FILLER              w  eh  l

% TAKE
take                t  ey  k

% PLEASE
please              p  l  iy  z

% OBJECT
pencil              p  ey  n  s  iy  l

% CALL
call                k  aa  l

% NAME
mohan               m  o  h  ax  n
```

The compilation returned no error. But the execution of the engine showed errors. The error was due to the untrained acoustic model for the written vocabulary. A temporary workaround was performed. The following command solved the issue but gave erroneous result

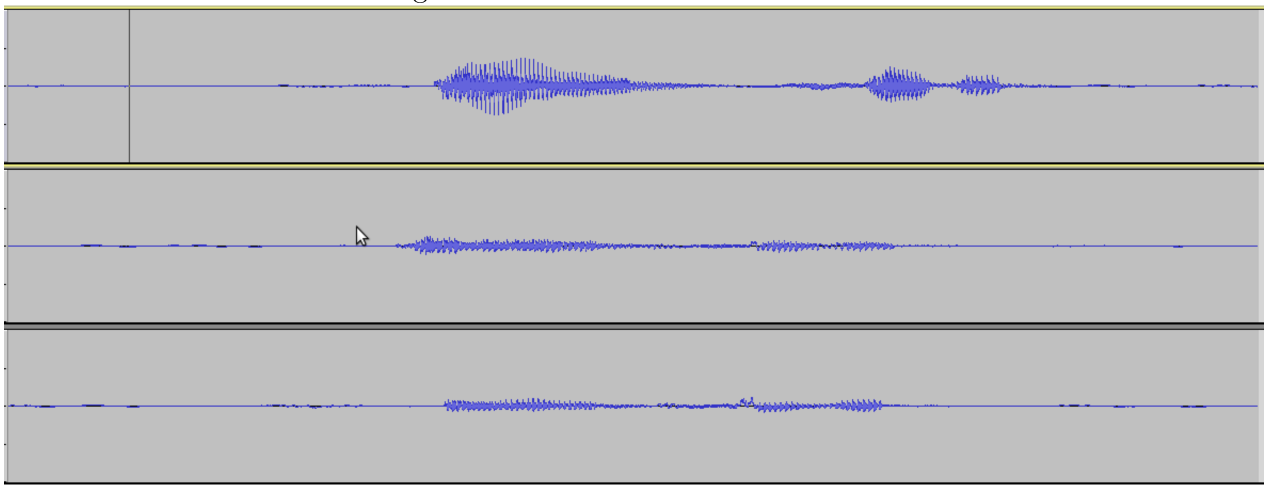
```
$ julius -input file -C rules.jconf -forcedict
```

The idea was given up and the final choice was to use the existing sample itself.

Testing the engine

4 words , *seven*, *eight*, *steve* and *maclean* were chosen for test purpose. 3 samples of each word were created, one in a typical american accent, one in a bass voice and one in a typical rural indian accent. The following are the waveforms of the signals, picked up by *Audacity*

Figure 2: Wave forms for the word 7



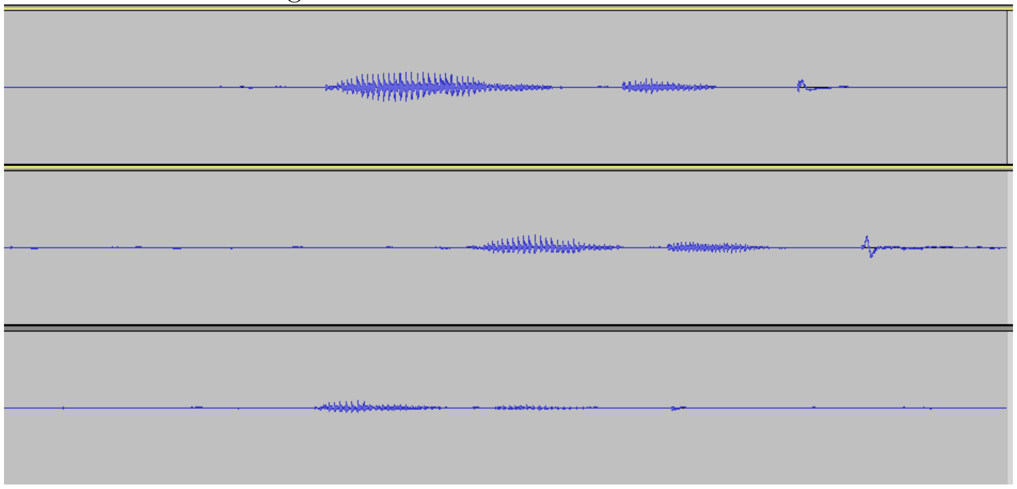
The word *seven* has a relatively clear pronounciation, hence was less erroneous.The phonetic reperesentation of seven is *s eh v ax n*.

The output in the 3 cases were as follows

```
<s> DIAL SEVEN </s>
<s> DIAL SEVEN </s>
<s> DIAL ZERO </s>
```

Clearly, the output is very good. The last case was the signal of an indian voiced person.

Figure 3: Wave forms for the word 8

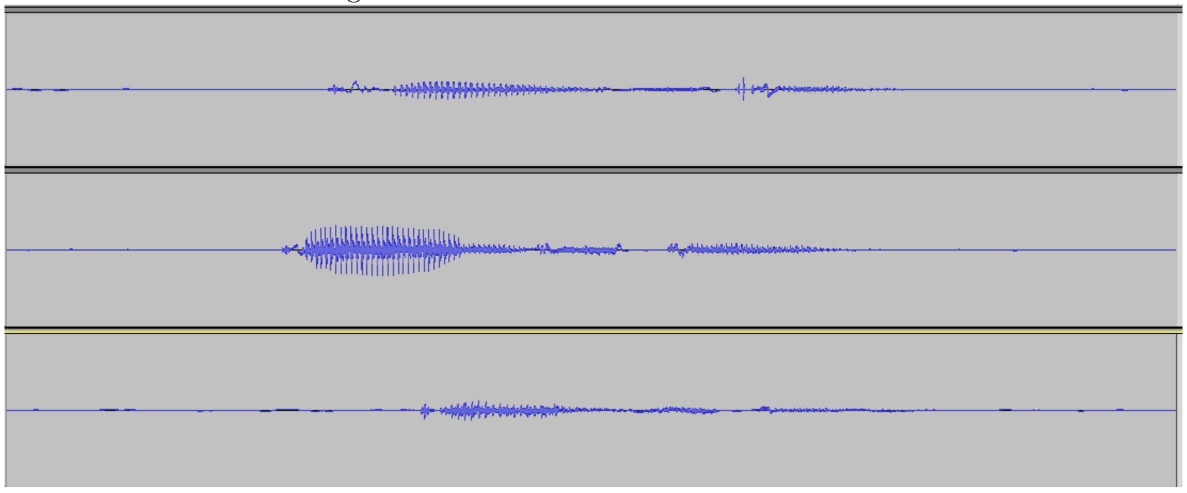


The word *eight* was did not give satisfactory result. The pronounciation is a little confusing.The phonetic translation is *ey t*. It is to be realised that the word 8 is confused for many other word even by humans. Hence the result is as expected.

```
<s> DIAL THREE </s>
<s> DIAL THREE </s>
<s> DIAL EIGHT </s>
```

In the above case, the indian voice gave the correct result.

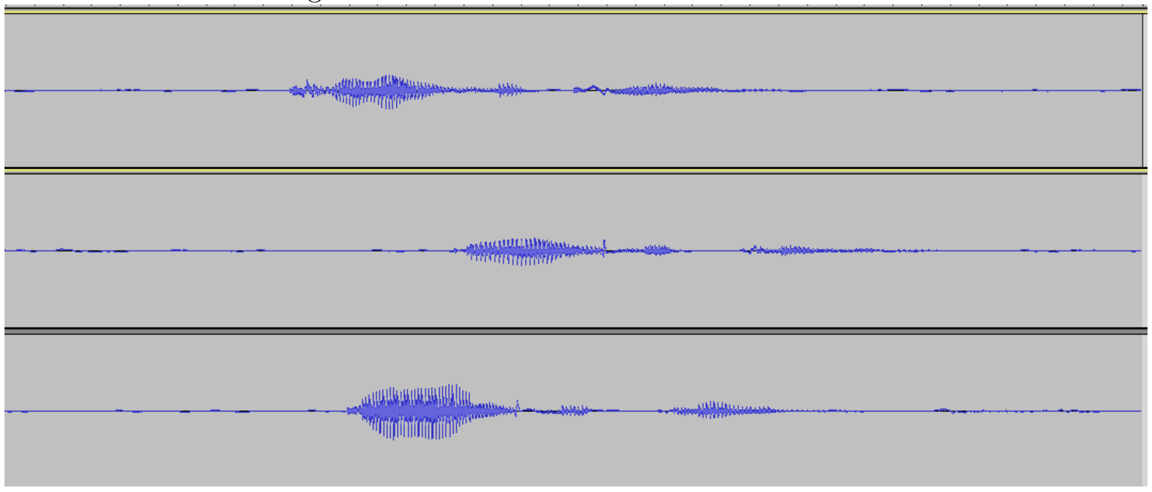
Figure 4: Wave forms for the word steve



The word *steve* gave a similar output to *eight*.Its phonetic translation is *s t iy v*. The result was as follows

```
<s> CALL STEVE </s>
<s> PHONE STEVE </s>
<s> PHONE STEVE </s>
```

Figure 5: Wave forms for the word maclean



The word *maclean* gave a very impressive result. The phonetic translation is *m ax k l ey n*. Due to its distinct and clear pronounciation, the computer recognised it with much ease. The output was as follows
<s> CALL MACLEAN </s>
<s> PHONE MACLEAN </s>
<s> CALL MACLEAN </s>

Immediate inference

The results of the tests were not very deviant. The results were very much close to the expected output. It is clear that the acoustic model included was made for the american accent of english. However, in few cases, the indian accent worked better. Also to be noted is that very close sounding words are misinterpreted by the computer. For example, the words *call* and *phone* sound very similar and are hence a source of error. The names used had no error. The grammar was very little and hence the detection was smooth. The inclusion of a large vocabulary will decrease the success rate of the engine.

Conclusion

The following was concluded from the test performed

- The neural network approach to speech detection is the best approach compared to the traditional computer algorithmic approach. This gave a much control over the probabilistic combinations.
- A flexible and trainable speech engine is a better option compared to the close source speech engine, since the engine can be trained for any language, irrespective of the complexity.
- The julius engine's test gave very interesting conclusions. Similar sounding words were found to be erroneous. This can be corrected by training the acoustic model better
- One more issue to solve is the inclusion of different types of pronounciations. A possible workaround is writing a more inclusive vocabulary file. For example consider the name *mohan*. To account for various pronounciations, the vocabulary can be written as follows

  ```
  %NAME
  mohan m ao h ax n
  mohan m oh ax n
  mohan m ao ax n
  ```

  This inclusion of different pronounciation will take care of all possible human calls for the name mohan.
- The speech signal is also to be cleared of any possible noise in the background. Any noise present will try to make the output erroneous. The possible solution does not lie in the software but we need to have better hardware to cancel the ambient and background noise.

The overall conclusion is that for the speech engine to accurately process and output the speech, the computer needs to be trained for all possibilities, we cannot assume anything.

Acknoweledgements

I would like to thank all my friends, relatives, and professors for helping me write this article. I would like to include few special mentions without whose help the work would remain incomplete.

- My wing mates in the hostel, who provided me with mic and other hardware when required
- My brother, who helped me to edit the sound signals and use them
- Professor Dr.Chandrashekhar of CS Department, IITM who gave a very informative lecture on speech recognition
- Last but not the least, Professor Sreesh Chaudhary, who tirelessly guided me throughout the work and helped me to design the article

References

[1] *www.google.com*

[2] *www.wikipedia.org*

[3] *www.learnartificialneuralnetwors.com*

[4] Lectures on linguistics by *Dr.Sreesh Chaudhary*

[5] Open source speech engine *julius*

[6] *www.voxforge.com*