Media Engineering and Technology Faculty
German University in Cairo

Speech Recognition and Pronunciation Evaluation for Quran

Bachelor Thesis

Author:	Abdelrahman Hamed
Supervisors:	Dr. Mohamed Elmahdy
Submission Date:	15 May, 2016

Media Engineering and Technology Faculty
German University in Cairo

Speech Recognition and Pronunciation Evaluation for Quran

Bachelor Thesis

Author: Abdelrahman Hamed
Supervisors: Dr. Mohamed Elmahdy

Submission Date: 15 May, 2016

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Abdelrahman Hamed
15 May, 2016

Acknowledgments

Firstly, I would like to thank professor Mohamed Elmahdy for giving me the chance to work on this project under his supervision and also thank him for all the help and the support he gave me which made this project more than beneficial. I also would like to thank my family for supporting me through out my years in the GUC and than all my friends who helped me in this project.

Abstract

A large number of Muslims are having difficulty pronouncing and learning the Quran. This project's first aim is to help making the learning easier by converting the spoken Aya from any Sura from wave form to text form to help the user to read and understand certain words that could not be heard appropriately. The process of converting from wave form to text form is called Automatic Speech Recognition (ASR). ASR is recognizing, identifying sentences or just words said in some specific language on a machine. It is basically "identifying what is being said" by converting the speech to a readable text. The second aim for this project is evaluating the pronouncing of the Quran by giving the user the score whether the Aya was said correctly or not this process is called Pronunciation Evaluation and it is done by evaluating the sentences whether they are being said correctly or not. The application used in this project are CMU Pocketsphinx and CMU Sphinx3aligner.

Contents

Chapter 1

Introduction

1.1 Motivation

Speech recognition has been developed through the years and it is being used in many applications in our lives like Siri and Cortana. knowing that speech recognition reached this far in many applications and in many languages, it encouraged to make a speech recognition API work for Arabic too specifically "The Holy-Quran". Meaning that when someone is reading or inputting an audio file (Aya from any Sura) it should display what is being said. Pronunciation evaluation is an essential method to help the user to learn the correct pronunciation of the Quran which evaluates the input audio to stored data to check if the pronouncing is correct. In this project, systems were created to recognize Arabic speech or in this case Quran speech and to evaluate it.

1.2 Thesis Aim

The main aim of this thesis is to create a system capable of recognizing the reading of any Aya from the Holy-Quran and convert it to readable text using only simple command line, also the system should be able to evaluate the correctness of the sentences being said.

1.3 Thesis Structure

In this thesis the work will be done as in the second chapter more about speech recognition theoretical background will be discussed along with the Acoustic Models (AMs), Language Model (LM), Phonetic Dictionary and the pronunciation evaluation. In the third chapter the process and the design of the systems will be discussed in further details and more illustration, and also the problems faced and their solutions. In the last part is the conclusion of the work done in this thesis and the improvements and the possibility of future work that can be made.

Chapter 2

Background

Automatic speech recognition (ASR) is a system used to recognize the spoken languages and also understand them. ASR is built by assigning or mapping a specific signal with to a specific word or sentence this is for the recognition part, for the understanding part the Automatic speech understanding (ASU) is used to produce some sort of understanding for the words and the sentences. ASR was and is being used in many fields such as telephony as the system can recognize some words like ”yes” or some commands like ”Acounnting please”, in human computer interaction, in dictation and many other fields. ASR is considered much simpler and better in handling the commands in many applications than visual interface in manipulating and controlling the objects. The ASR system are normally made as the next Figure [1][5].

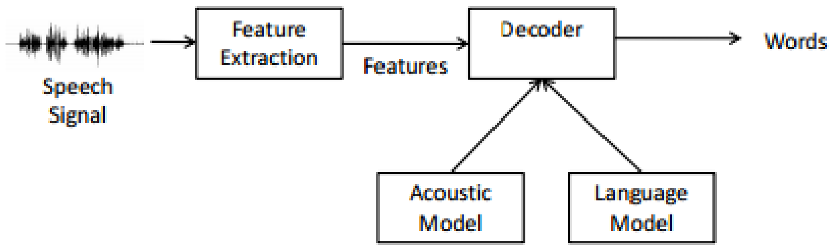


Figure 2.1: ASR System

2.1 Dimensions of Variation

One of the most important parameters of the ASR in order to work in the most efficient way are the dimensions of variation. ASR has four dimensions that will be discussed in this thesis [1].

2.1.1 First Dimension: Vocabulary Size

Vocabulary size is one of the most effective dimension in the ASR because, it affects the easiness of the recognition, for example the size of the vocabulary in **human-to-human** conversation or in broadcast news (range of 64,000 words) is much harder to recognize than the vocabulary of size 2 such as ”yes” or ”no” .The variance of the difficulty tends to the number of distinct word in the vocabulary so the system can organize the word like digits so it is easier to organize 2 words than 64,000 words and this is called **digits task** where the system deals with each word as a digit [1].

2.1.2 Second Dimension: Fluent Speech

Fluent speech helps a lot in creating the perfect recognition system where the system work as it recognize the **Isolated words** so the meaning of fluent speech is that every word must be followed by silence or a pause, clear and slow. Therefor the recommended to use speeches which are **human-to-machine** speeches because they are slow, loud, clear and isolated other than **human-to-human** conversation or a broadcast news [1].

2.1.3 Third Dimension: Channel and Noise

The main idea of this dimension is the recognition of any speech is much easier if the record of any speech is done in a place quite and don't have any kind of noise and also the microphone used in recording is a mounted head one [1].

2.1.4 Fourth Dimension: The Accent of The Speaker

The accent of the speaker that recorded the speech is an important factor in the speech recognition process. The more the spoken language was spoken in standard accent or one that similar to the language that the system was trained on the more the recognition became easier other than foreign languages or incomprehensible speech for example speech recorded from small children that is so much harder to recognize [1].

A study has been done in 2001 shows the word error rate (WER) in the recognition is higher 3 to 4 times in case of Japanese-accented or Spanish accented English than the standard English the results are in Table 2.1

Table 2.1: Error rate results on different tasks [1]

Task	Vocabulary	Error Rate %
TI Digits	11 (zero-nine, oh)	0.5
Wall Street Journal read speech	5,000	3
Wall Street Journal read speech	20,000	3
Broadcast News	64,000+	10
Conversational Telephone Speech (CTS)	64,000+	20

2.2 Speech Recognition Architecture

The speech recognition is a probability equation where the system take input speech signal and sentence the system should outputs the high probability sentence lets assume that the system takes speech signal **O** and a sentence **W** then the output $\hat{W}$ should be expressed by:

$$\hat{W} = argmax_{W \in L} P(W|O) \tag{2.1}$$

where L is the Language that contains all the sentences [5]. the modified version:

$$\hat{W} = argmax_{W \in L} P(O|W)P(W) \tag{2.2}$$

2.2.1 Feature Extraction

The process of transforming the speech signal to a **Feature Vector** is called the **Feature Extraction**, this process is done by many methods but the one that will be discussed in this thesis is the **Mel-Frequency Cepstal coefficient**(**MFCC**) technique [1]. As shown in the below figure this technique is done on several stages:

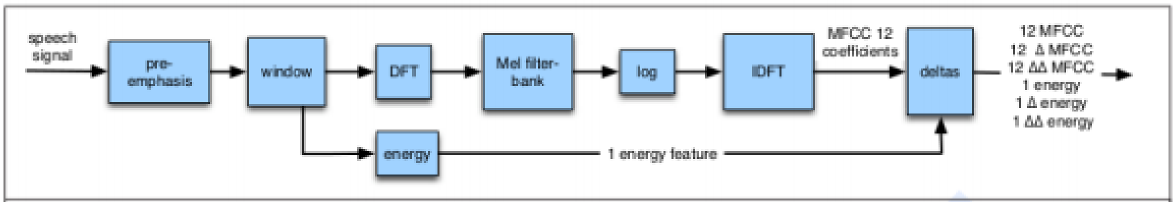


Figure 2.2: Feature Extraction Stages

2.2.1.1 Preemphasis

Preemphasis is the process that applies filter to the wave file in order to enhance the energy of the wave file and that increases the amount of the vocabulary available to the acoustic model which improves the accuracy of the phonemes detection [1]. The next figure shows the accuracy before and after applying the Preemphasis.

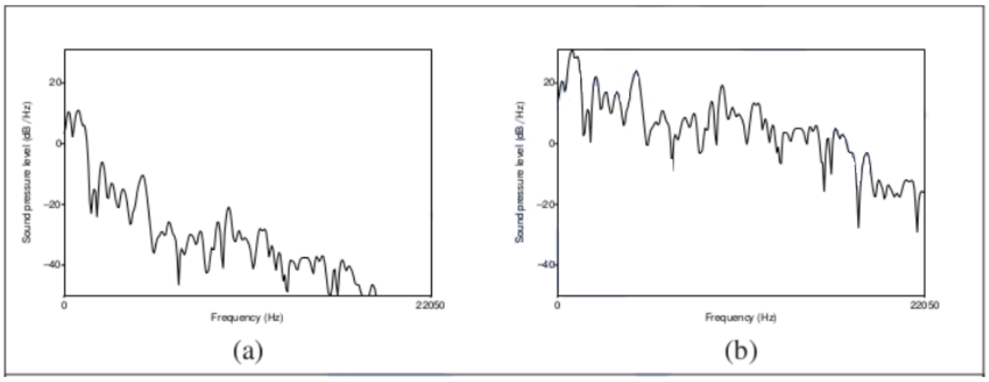


Figure 2.3: The Effect Of Applying Preemphasis[1]

.

2.2.1.2 Windowing

Windowing is the process of applying band-bass filter that gives us the signal in some period and the other periods are zeros then the features are extracted from the window area. The reason why Windowing is applied to wave file because the speech's statistical properties is changing in non-stop rate which makes it non-stationary signal which makes it harder to extract the features, so after applying the Windowing the window area is becoming small enough to represent certain phoneme. Knowing that the window is so small it is considered to be stationary signal which makes the features extraction much easier [1].

2.2.1.3 Discrete Fourier Transform

The next step after applying Windowing is to extract the feature and this is done by applying Discrete Fourier Transform (DFT) to know how the energy of the signal varies with the frequency band. DFT can be done by a pre-implemented algorithm called Fast-Fourier Transform (FFT) [1].

2.2.1.4 Mel-filter Bank

Because all humans do not hear or sense frequencies above 1000 Hz the Mel-filter Bank is applied to remove and filter the wave file from any frequencies that humans are less sensitive to and in doing so the efficiency of the acoustic model is improved greatly [1].

2.3 Models

Speech recognition applications normally use three models in order to recognize in correct way these models are **Language Model**, **Acoustic Model** and **Phonetic Dictionary** [2]. As discussed before in the modified version of the probability equation there was P(W) and this part is handled by the Acoustic model and the P(W|O) is handled by the Language model all of this is will be discussed in the next sections [5].

2.3.1 Language Model

The simple way to describe the language model is a collection of unique words in the speech text and the possible combinations of the words that appeared in the speech text. All language models are using **N-gram** which a simple sequence of N items from any speech those items could be words, phones , letters or even syllables those items are divided according to the N-number if N=1 it is called uni-gram, if N=2 it is called bi-gram, if N=3 it is called tri-gram, if N=4 it is called four-gram and so on. The difference between each one of them is the way to divide each word and each sentence for example " for word three in uni-gram is divided to t,h,r,e,e but in bi-gram th,hr,re,ee" and so on [2][1].

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{k-N+1}^{k-1}) \tag{2.3}$$

The language models also use finite state automation with wight. In order to get better accuracy the language model must be very successful in predicting the next word. Language model can be also described by sentence W consists of (w1,w2,w3,w4...wN), so the language model can be expressed as:

$$P(W) = P(w_1, w_2, w_3, ..., w_n) \tag{2.4}$$

2.3.2 Acoustic model

As said before the Acoustic model is the handler of the P(W|O) of the probability equation. The Acoustic model contains all the phonemes for the language that can be recognized, it is called Statistical modeling and can be preformed on all kinds of phonemes by method called **Hidden Markov model** (**HMM**).Each model has five states because each phoneme has three sub phones, the five states are start state,beginning state, middle state, finish state, end state [5].

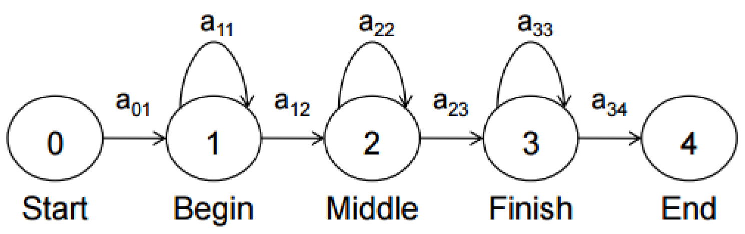


Figure 2.4: Acoustic Model States

Because some phonemes have different acoustic proprieties in each word, regarding this there are two modeling methods the first one is called **Context Independent** (**CI**) this method does not take into consideration the phonemes prior or after the modelled phoneme, the other method is called **Context Dependent** (**CD**) this method takes into consideration the context of the phonemes [5].

2.3.3 Phonetic Dictionary

Phonetic Dictionary contains the each word in the speech mapped to its right phones but in some cases this mapping is not effective because there is some words that could have more than one phone or some different word that could have the same phones, there is also other different type of mapping like machine learning algorithm but this is not in the range of the study for this project [2].

2.3.4 Evaluation

In order to evaluate the speech recognition systems there are two standard metrics the first is called called **The Word Error Rate**(**WER**) the evaluation is based on how much the assumed word is different from the correct word in the transcription files of the speech. There are three types of error

- **Word Substitution** the recognizer miss-understands a word by matching a word to another word.
- **Word Insertion** : the recognizer adds an extra word to its assumption that was not in the speech segment
- **Word Deletions** the assumption is lacking a word that was in the speech segment

The WER is calculated from those three

$$WER = \frac{Insertions + Deletions + Substitutions}{Total\ Words\ in\ Transcript} \times 100 \tag{2.5}$$

The second standard metric is called **Sentence error rate** (**SER**) the evaluation is based on how many sentences had at least one error and it is calculated by the equation

$$WER = \frac{Number\ of\ sentences\ with\ at\ least\ 1\ error}{Total\ number\ of\ sentences} \times 100 \tag{2.6}$$

2.4 CMU Sphinx Toolkit

CMU Sphix is the toolkit used in this project. This kit consists of four tools the first one is **sphinxbase** this tool contains the essential libraries for the the other tools (**pocketsphinx** and **sphinxtrain**) as well as the ability to manipulate acoustic feature and audio files, the second one is the **sphinxtrain** the function of this tool is training the language model, the phonetic dictionary, transcription files and the wave files to get an acoustic model that can recognize the speech, the third tool is the **pocketshpinx** the main function for this tool is to recognize the speech by calling the acoustic model, the language model and the phonetic dictionary then outputs the speech in text , the last tool is **sphinx3aligner** this tool is used to get the error rate of the pronunciation [3].

2.5 CMU SLM Toolkit

This tool kit is used to generate the language model for the Holy-Quran by taking the whole text file of the Quran and convert it to language model [8] that was used in **pocketsphinx** tool using small steps that will be discussed in the implementation chapter.

2.6 Quran Aligner

Quran Aligner is an application used for automatic long audio alignment and confidence scoring for conversational Arabic speech in this project the application was used for Quran segmentation and scoring instead of conversational Arabic speech this process is by a method called split and merge approach. A basied LM is trained and pronunciation model (PM) is utilized. The alignment is done on two passes the first one creates AM to recognize the Quran speech and the second pass AM adaptation is applied to improves the recognition [6]. After the segmentation the output will be used in the CMU Sphinx Toolkit.

2.7 Sphinx3 Aligner

Sphinx3 Aligner is a toolkit used for taking the transcription file of a wave file and determine where in time exactly a specific word will occur it is basically the opposite of speech recognition process because it take text format and convert it to wave format [7]. In this project the Sphinx3 Aligner was used to produce acoustic scores for certain readings that shows the user the percentage of the correct reading said [4].

Chapter 3

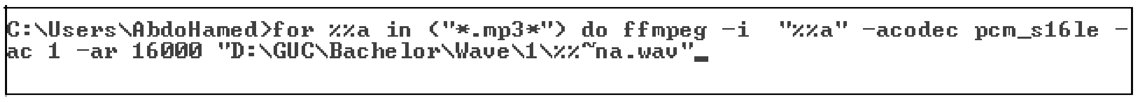
Implementation

3.1 First Attempt

3.1.1 First Stage: Collecting Data

Since this project is about the Quran and making speech recognition and pronunciation evaluation for the Quran the first goal was to collect huge amount of data for Quran readings, the target for this project was to get the whole Quran wave files (114 Sura) to a ten different Quran readers. The ten readers were different nationalities in order to get the maximum variety of the sounds, also the readings were Tartel version not Tagwed version in order to make the process much faster and more efficient.

After downloading all readers, some replacement had to be done in order to get the most suitable readings because there were some reads that their readings was recorded over radio or had some echo which did not work in the testing stage.Knowing that the applications that were used deal only with wave files with 16KHz sample rate, mono channel and have the extension .wav a program called ffmpeg was used to convert the reading s from mp3 to 16KHz mono channel wav files [7]. To do that simple command was used in each folder of the readings that were downloaded:



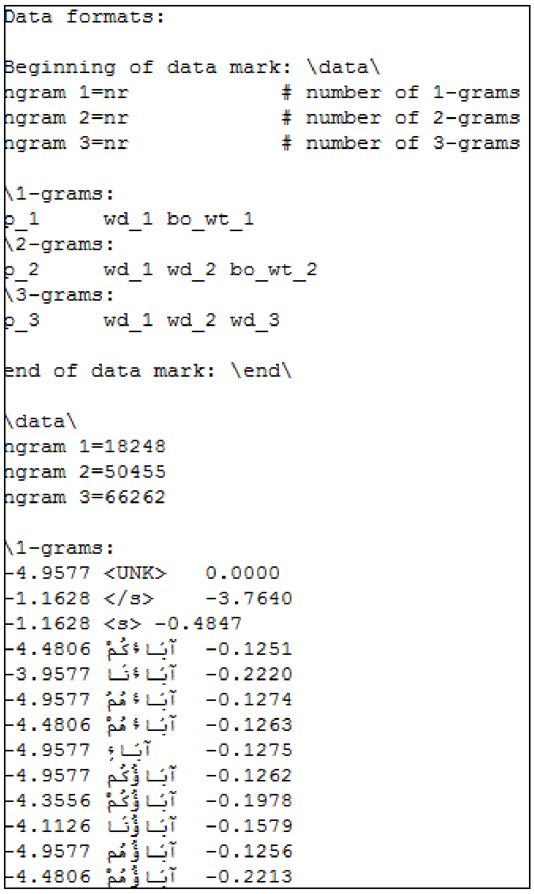
```
C:\Users\AbdoHamed>for %%a in ("*.mp3*") do ffmpeg -i  "%%a" -acodec pcm_s16le -
ac 1 -ar 16000 "D:\GUC\Bachelor\Wave\1\%%~na.wav"
```

Figure 3.1: Command for converting mp3 files to wav files with simple rate 16KHz mono channel

There was problem that the readings were downloaded each Sura with it‘s name for example Sura one was 01-Alfathea so a script in **Java** was created to rename the files in a three bit counter 000,001,002...114, then a text file that contains the whole Quran was divided to 114 text files each one contains the Sura that belongs to that number the numbering of the files was also in three bit counter.

3.1.2 Second Stage: Creating The Language Model

The file that contains the whole Quran text was used in creating the Language model using the application CMU SLM Toolkit, it takes the Quran text and with small command it convert the text file of the Quran to the language model in extension .arpa that will be used later on the Pocktshpinx toolkit.

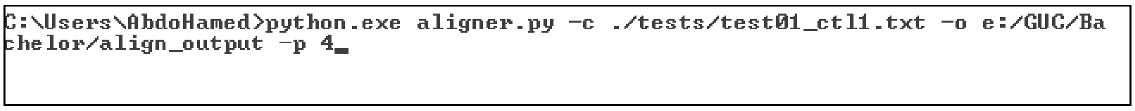


```
Data formats:

Beginning of data mark: \data\
ngram 1=nr             # number of 1-grams
ngram 2=nr             # number of 2-grams
ngram 3=nr             # number of 3-grams

\1-grams:
p_1     wd_1 bo_wt_1
\2-grams:
p_2     wd_1 wd_2 bo_wt_2
\3-grams:
p_3     wd_1 wd_2 wd_3

end of data mark: \end\

\data\
ngram 1=18248
ngram 2=50455
ngram 3=66262

\1-grams:
-4.9577 <UNK>   0.0000
-1.1628 </s>    -3.7640
-1.1628 <s> -0.4847
-4.4806 آبَاءُكُمْ   -0.1251
-3.9577 آبَاءَنَا   -0.2220
-4.9577 آبَاءَهُمْ   -0.1274
-4.4806 آبَاءُهُمْ   -0.1263
-4.9577 آبَاءِ     -0.1275
-4.9577 آبَاؤُكُم   -0.1262
-4.3556 آبَاؤُكُمْ   -0.1978
-4.1126 آبَاؤُنَا   -0.1579
-4.9577 آبَاؤُهُم   -0.1256
-4.4806 آبَاؤُهُمْ   -0.2213
```

Figure 3.2: Sample of ARPA LM file

3.1.3 Third Stage: Quran Segmentation

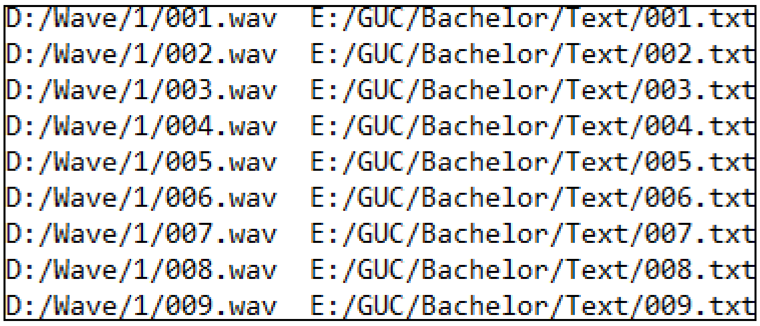
At this point the Quran Aligner was used to get the segmentation of the Quran by creating control files that where created by hard coded scripts using also **Java**. Those control files contain the location for each Sura from each reader and the location of the corresponding text of it and inputs them to the Quran aligner. The Quran aligner takes each Sura's wave file and file text and start to align them to get the segments, the command to start the process has two parts the first part is the location of the control file and the second part is the location for the output folder.



```
C:\Users\AbdoHamed>python.exe aligner.py -c ./tests/test01_ctl1.txt -o e:/GUC/Ba
chelor/align_output -p 4_
```

Figure 3.3: Command for calling the control files that will run the aligner.py (Quran Aligner)

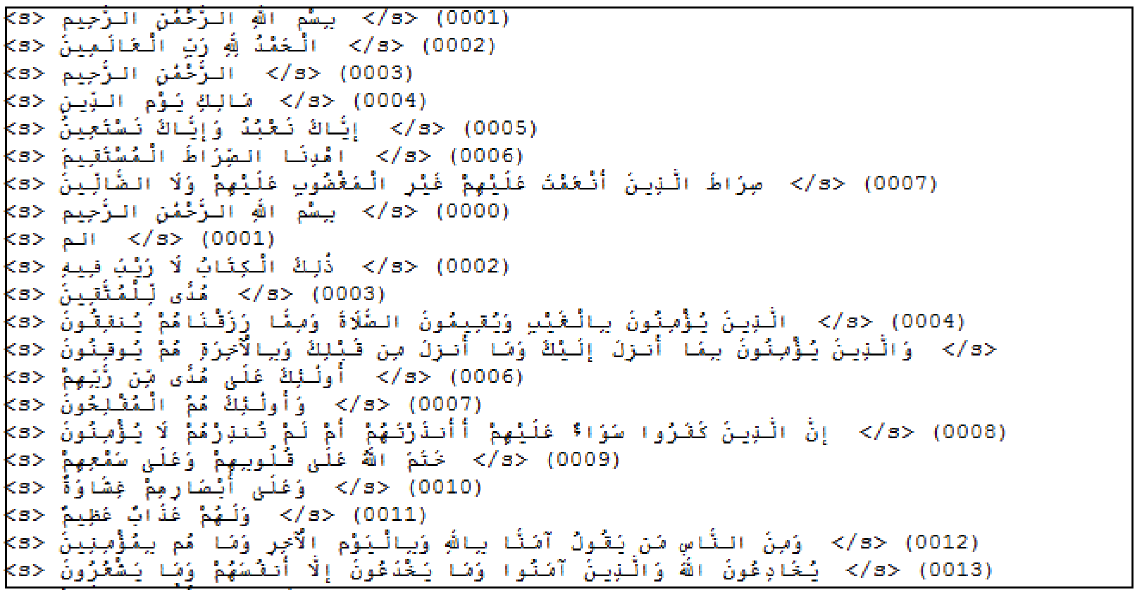
For the control file has the pattern of lines each line has two parts the first part is the location of the wave file and the second part is the location of the text file.



```
D:/Wave/1/001.wav  E:/GUC/Bachelor/Text/001.txt
D:/Wave/1/002.wav  E:/GUC/Bachelor/Text/002.txt
D:/Wave/1/003.wav  E:/GUC/Bachelor/Text/003.txt
D:/Wave/1/004.wav  E:/GUC/Bachelor/Text/004.txt
D:/Wave/1/005.wav  E:/GUC/Bachelor/Text/005.txt
D:/Wave/1/006.wav  E:/GUC/Bachelor/Text/006.txt
D:/Wave/1/007.wav  E:/GUC/Bachelor/Text/007.txt
D:/Wave/1/008.wav  E:/GUC/Bachelor/Text/008.txt
D:/Wave/1/009.wav  E:/GUC/Bachelor/Text/009.txt
```

Figure 3.4: Sample of the Control file

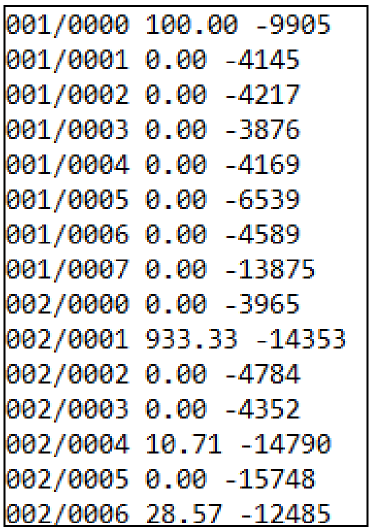
The Quran aligner‘s second output is the transcription files, the transcription file it a text file that contains each segment started with ”<s>” and ended with ”</s>” and the number of these segments next to them for example:



```
<s> بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ </s> (0001)
<s> الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ </s> (0002)
<s> الرَّحْمَٰنِ الرَّحِيمِ </s> (0003)
<s> مَالِكِ يَوْمِ الدِّينِ </s> (0004)
<s> إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ </s> (0005)
<s> اهْدِنَا الصِّرَاطَ الْمُسْتَقِيمَ </s> (0006)
<s> صِرَاطَ الَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ الْمَغْضُوبِ عَلَيْهِمْ وَلَا الضَّالِّينَ </s> (0007)
<s> بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ </s> (0000)
<s> الم </s> (0001)
<s> ذَٰلِكَ الْكِتَابُ لَا رَيْبَ فِيهِ </s> (0002)
<s> هُدًى لِلْمُتَّقِينَ </s> (0003)
<s> الَّذِينَ يُؤْمِنُونَ بِالْغَيْبِ وَيُقِيمُونَ الصَّلَاةَ وَمِمَّا رَزَقْنَاهُمْ يُنْفِقُونَ </s> (0004)
<s> وَالَّذِينَ يُؤْمِنُونَ بِمَا أُنْزِلَ إِلَيْكَ وَمَا أُنْزِلَ مِنْ قَبْلِكَ وَبِالْآخِرَةِ هُمْ يُوقِنُونَ </s>
<s> أُولَٰئِكَ عَلَىٰ هُدًى مِنْ رَبِّهِمْ </s> (0006)
<s> وَأُولَٰئِكَ هُمُ الْمُفْلِحُونَ </s> (0007)
<s> إِنَّ الَّذِينَ كَفَرُوا سَوَاءٌ عَلَيْهِمْ أَأَنْذَرْتَهُمْ أَمْ لَمْ تُنْذِرْهُمْ لَا يُؤْمِنُونَ </s> (0008)
<s> خَتَمَ اللَّهُ عَلَىٰ قُلُوبِهِمْ وَعَلَىٰ سَمْعِهِمْ </s> (0009)
<s> وَعَلَىٰ أَبْصَارِهِمْ غِشَاوَةٌ </s> (0010)
<s> وَلَهُمْ عَذَابٌ عَظِيمٌ </s> (0011)
<s> وَمِنَ النَّاسِ مَنْ يَقُولُ آمَنَّا بِاللَّهِ وَبِالْيَوْمِ الْآخِرِ وَمَا هُمْ بِمُؤْمِنِينَ </s> (0012)
<s> يُخَادِعُونَ اللَّهَ وَالَّذِينَ آمَنُوا وَمَا يَخْدَعُونَ إِلَّا أَنْفُسَهُمْ وَمَا يَشْعُرُونَ </s> (0013)
```

Figure 3.5: Sample of the Transcription file

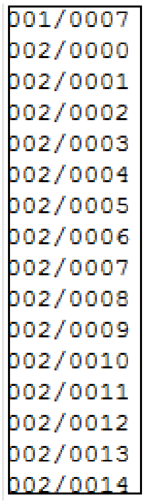
The third output is the error rate of each segments and the less the number in the middle the more the segment was accurate and the number on the right is the confidence scoring, this number varies with the length of the segment itself.



```
001/0000 100.00 -9905
001/0001 0.00 -4145
001/0002 0.00 -4217
001/0003 0.00 -3876
001/0004 0.00 -4169
001/0005 0.00 -6539
001/0006 0.00 -4589
001/0007 0.00 -13875
002/0000 0.00 -3965
002/0001 933.33 -14353
002/0002 0.00 -4784
002/0003 0.00 -4352
002/0004 10.71 -14790
002/0005 0.00 -15748
002/0006 28.57 -12485
```

Figure 3.6: Sample of the Error Rate file

The fourth output is the fileids it a text file contains the numbers of segments in each Sura



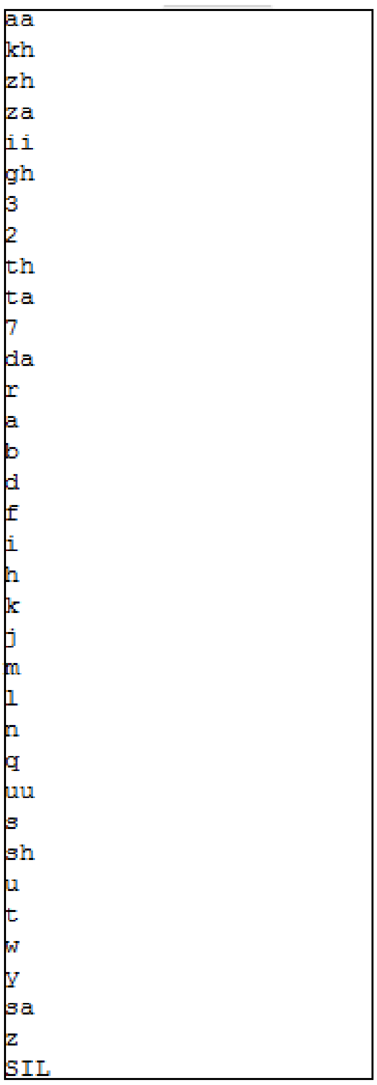
```
001/0007
002/0000
002/0001
002/0002
002/0003
002/0004
002/0005
002/0006
002/0007
002/0008
002/0009
002/0010
002/0011
002/0012
002/0013
002/0014
```

Figure 3.7: Sample of the Fileids file

3.1.4 Fourth Stage: Training The Acoustic Model

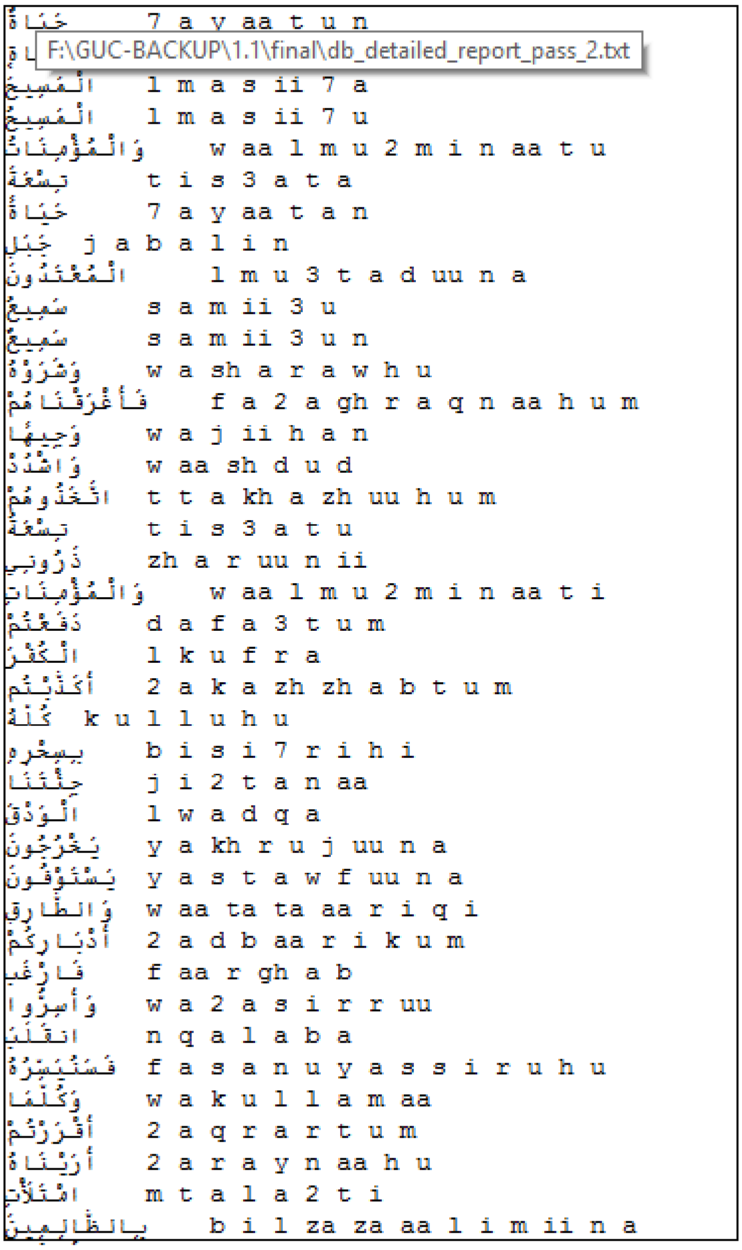
After aligning the ten Quran readers, the transcription files of the first eight readers were put to one file called "Quran_train.transcription" and the fileids of the same eight

readers were put to one file called ”Quran_train.fileids”. The same process was applied to the last two readers but the names of the files were ”Quran_test.transcription” and ”Quran_test.fileids”. The next step was initializing the CMU Pocketsphinx workspace, then adding the files.
•Quran_train.transcription.
•Quran_test.transcription.
•Quran_train.fileids.
•Quran_test.fileids.
•Quran.phone(taken from previous bachelor).
•Quran.dic(taken from previous bachelor) .
•Quran.filler(created automatically when the workspace was created) .



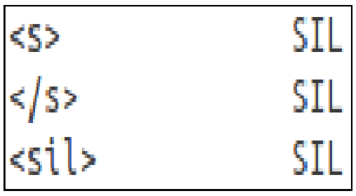
```
aa
kh
zh
za
ii
gh
3
2
th
ta
7
da
r
a
b
d
f
i
h
k
j
m
l
n
q
uu
s
sh
u
t
w
y
sa
z
SIL
```

Figure 3.8: The list of all phonemems used in Quran.phone file



```
حَيَاةٌ        7 a y aa t u n
ة ل F:\GUC-BACKUP\1.1\final\db_detailed_report_pass_2.txt
الْمَسِيحَ     l m a s ii 7 a
الْمَسِيحُ     l m a s ii 7 u
وَالْمُؤْمِنَاتُ    w aa l m u 2 m i n aa t u
تِسْعَةَ       t i s 3 a t a
حَيَاةً        7 a y aa t a n
جَبَلٍ   j a b a l i n
الْمُعْتَدُونَ     l m u 3 t a d uu n a
سَمِيعُ        s a m ii 3 u
سَمِيعٌ        s a m ii 3 u n
وَشَرَوْهُ      w a sh a r a w h u
فَأَغْرَقْنَاهُمْ    f a 2 a gh r a q n aa h u m
وَجِيهًا       w a j ii h a n
وَاشْدُدْ      w aa sh d u d
اتَّخَذُوهُمْ    t t a kh a zh uu h u m
تِسْعَةُ       t i s 3 a t u
ذَرُونِي       zh a r uu n ii
وَالْمُؤْمِنَاتِ    w aa l m u 2 m i n aa t i
دَفَعْتُمْ      d a f a 3 t u m
الْكُفْرَ      l k u f r a
أَكَذَّبْتُمْ    2 a k a zh zh a b t u m
كُلُّهُ   k u l l u h u
بِسِحْرِهِ      b i s i 7 r i h i
جِئْتَنَا       j i 2 t a n aa
الْوَدْقَ      l w a d q a
يَخْرُجُونَ     y a kh r u j uu n a
يَسْتَوْفُونَ    y a s t a w f uu n a
وَالطَّارِقِ    w aa ta ta aa r i q i
أَدْبَارِكُمْ    2 a d b aa r i k u m
فَارْغَبْ      f aa r gh a b
وَأَسِرُّوا     w a 2 a s i r r uu
انقَلَبَ       n q a l a b a
فَسَنُيَسِّرُهُ   f a s a n u y a s s i r u h u
وَكُلَّمَا      w a k u l l a m aa
أَقْرَرْتُمْ     2 a q r a r t u m
أَرَيْنَاهُ     2 a r a y n aa h u
امْتَلَأْتِ     m t a l a 2 t i
بِالظَّالِمِينَ     b i l za za aa l i m ii n a
```

Figure 3.9: ScreenShot from Quran.dic File



```
<s>       SIL
</s>      SIL
<sil>     SIL
```

Figure 3.10: Quran.filler File

and the segmentation (wave files) for the Quran readers divided to two part training (first eight readers) and testing (last two readers). At this point all the data preparation was done and summary about the collected data was taken

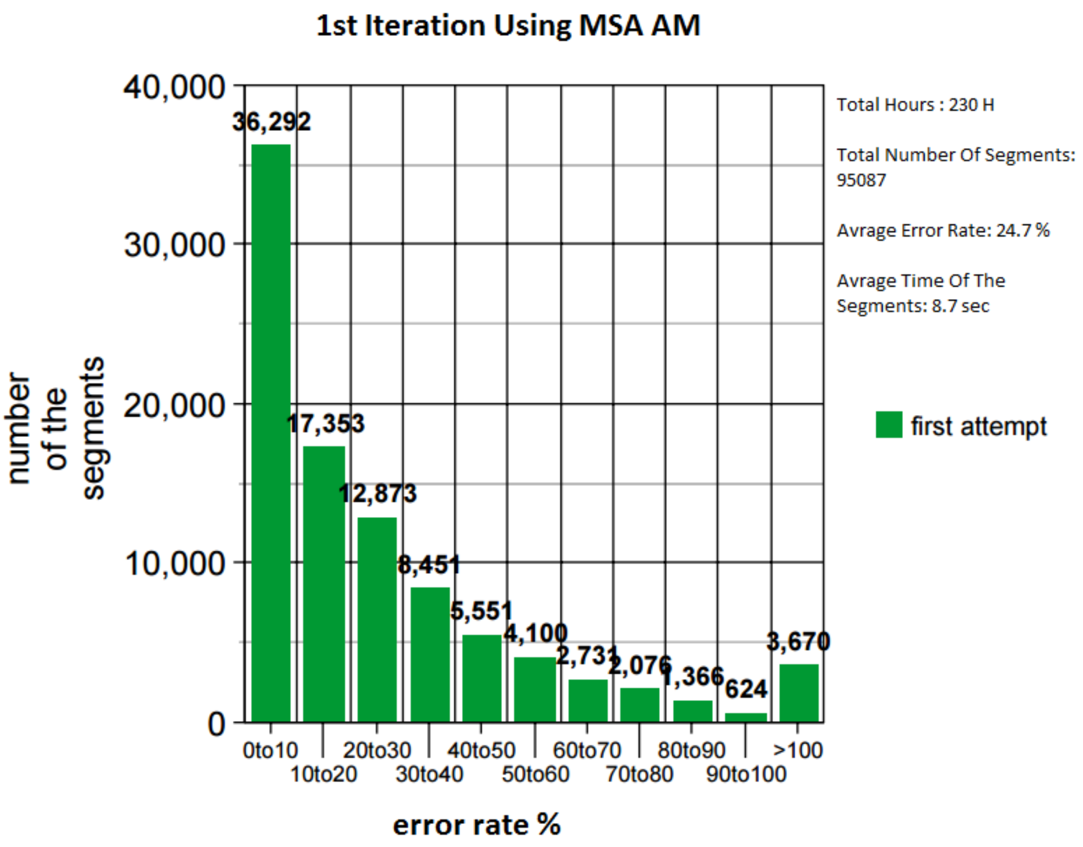


Figure 3.11: Collected data after aligning with Modern Standard Arabic Acoustic Model (MSA AM)

Due to some segments that had some noise and echo the training could not complete the training of the AM so the data was filtered according to the error rate of each segments by **Java** script that mainly loops on each output folder from the aligner and reads the files of the error rates and checks if the error rate exactly 0% the corresponding transcription and fileid was written to the new "Quran_train.transcription" and "Quran_train.fileids". After filtering the data the total number training segments was **25,711**, the total hour was **40 Hrs** and the Avg. Time was **16 sec.** and the training was completed successfully then the testing was started, but according to the huge amount of test segments and transcription that took hours and hours only a sample of testing data were used and the output was:

Table 3.1: Decoding(Testing) Results MSA AM

Parameter	Value
WER	12.4 %
ASR	27 %
Total Words	3261
Accuracy	52.87 %
Correct Word Percentage	67 %
Error Word Percentage	47.13 %
Substitution Errors	870
Deletion Errors	461
Insertion Errors	206

After that the AM files was taken to be used in the Quran Aligner and repeat the process again with the new AM.

3.2 Second Attempt

At this attempt the process of the first attempt was repeated but with the new AM files, also the first and the second stages were already done in the first attempt so there was no need to do them again and the third stage the AM files only that were replaced by the new files(Output of the Pocketsphinx), for the two Quran.phone and Quran.dic there was no change in them all the change was in the four file of the transcription and fileids.

The same steps were taken just like in the first attempt but the difference was that the alignment of the ten Quran readers was done with the new AM so the data the was selected to be trained was much bigger.

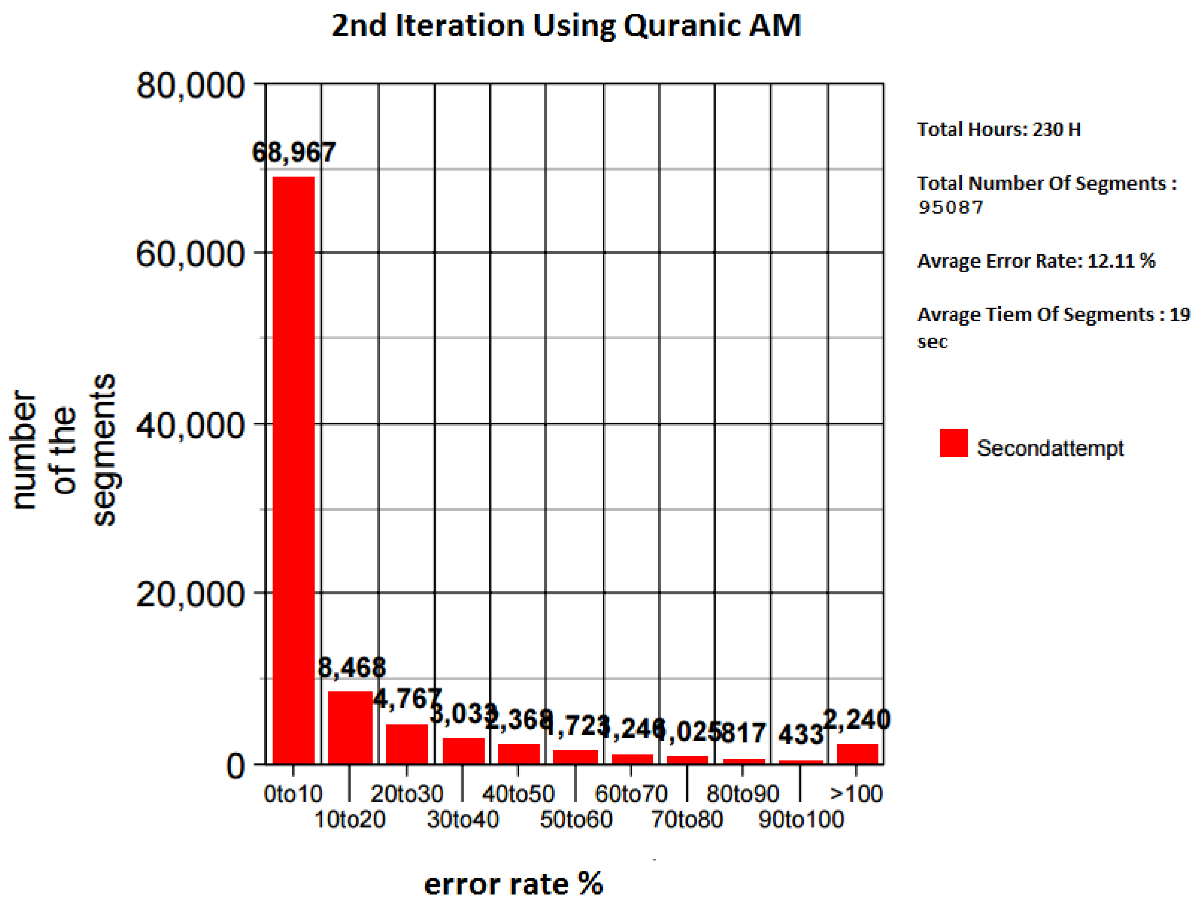


Figure 3.12: Collected data after aligning with Quranic Acoustic Model

Filter was applied as in the first attempt to get exactly 0% error rated segments and the number of segments was **58,943** and that is a huge number so a new filter had to be made, a **Java** script was made to filter the segments according to three parameters error rate, size and confidence score. This script checks the three parameters for each segments and filter them according to error rate exactly 0%, size less than 650KB and finally confidence score grater than -10000. After filtering the data the total number training segments was **33,029**, the total hour was **49 Hrs** and the Avg Time was **13.5 sec.**, the next step was training new AM and testing it and those were the results:

Table 3.2: Decoding(Testing) Results Quranic AM

Parameter	Value
WER	13.7 %
ASR	30.4 %
Total Words	2174
Accuracy	55.01 %
Correct Word Percentage	64.08 %
Error Word Percentage	44.99 %
Substitution Errors	673
Deletion Errors	108
Insertion Errors	197

3.3 Evaluating the Correctness Recorded Quranic speeches

At this point the speech recognition part is over and the next step was to evaluate the new acoustic model. In order to do that sphinx3 aligner was used by calling the files [4].

•The files of the Quranic AM.

•The phone File.

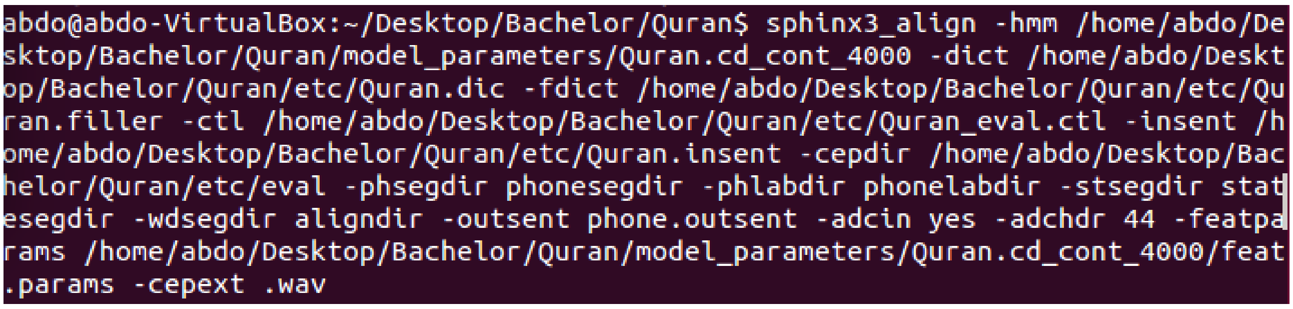
•The dictionary File.

•New file that has the fileids of the segments to be tested.

•New file that has the transcription of the segments to be tested.

•The wave files(segments) to be tested.

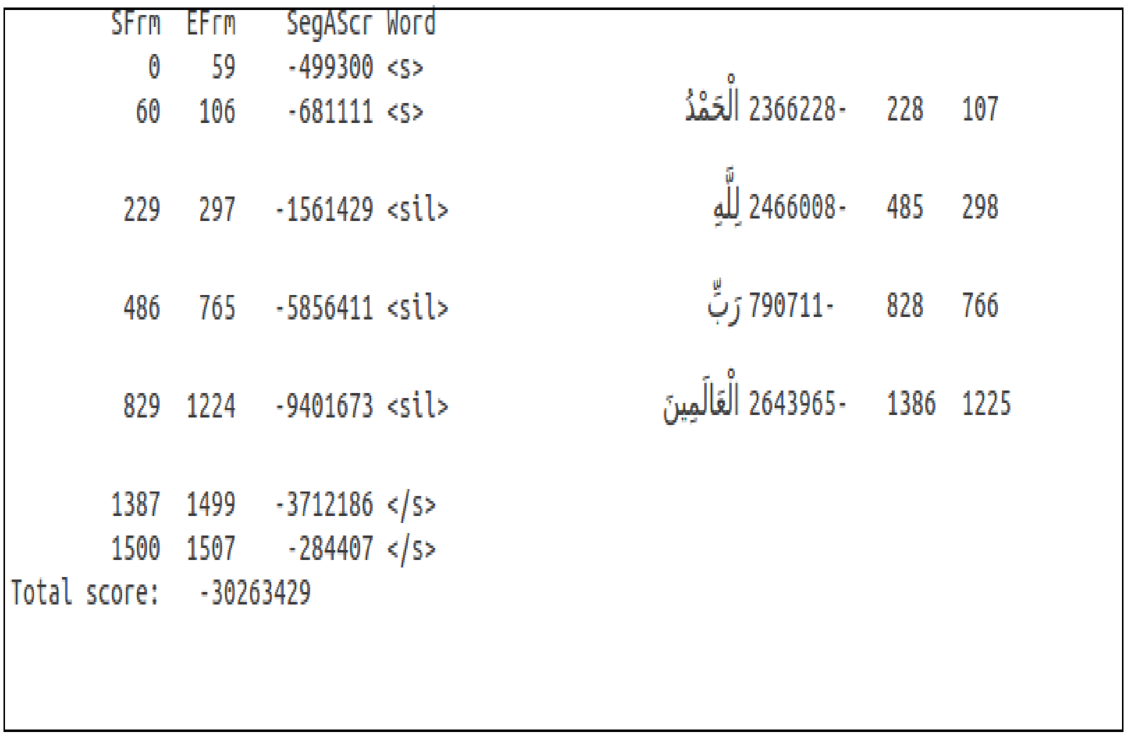
After collecting those data it is simple now all what is left is to run the following command to generate the acoustic scores for the new files.



```
abdo@abdo-VirtualBox:~/Desktop/Bachelor/Quran$ sphinx3_align -hmm /home/abdo/De
sktop/Bachelor/Quran/model_parameters/Quran.cd_cont_4000 -dict /home/abdo/Deskt
op/Bachelor/Quran/etc/Quran.dic -fdict /home/abdo/Desktop/Bachelor/Quran/etc/Qu
ran.filler -ctl /home/abdo/Desktop/Bachelor/Quran/etc/Quran_eval.ctl -insent /h
ome/abdo/Desktop/Bachelor/Quran/etc/Quran.insent -cepdir /home/abdo/Desktop/Bac
helor/Quran/etc/eval -phsegdir phonesegdir -phlabdir phonelabdir -stsegdir stat
esegdir -wdsegdir aligndir -outsent phone.outsent -adcin yes -adchdr 44 -featpa
rams /home/abdo/Desktop/Bachelor/Quran/model_parameters/Quran.cd_cont_4000/feat
.params -cepext .wav
```

Figure 3.13: Command for running sphinx3 aligner

After finishing the evaluation the output files that can be found in the **aligndir** folder each file has the total score of each segment and the more accurate the reading was the higher the score became. I nthe next figure example of the output file.



```
        SFrm  EFrm     SegAScr Word
           0    59     -499300 <s>
          60   106     -681111 <s>                        الْحَمْدُ 2366228-    228    107


         229   297    -1561429 <sil>                        لِلَّهِ 2466008-    485    298


         486   765    -5856411 <sil>                         رَبِّ 790711-    828    766


         829  1224    -9401673 <sil>                   الْعَالَمِينَ 2643965-    1386  1225


        1387  1499    -3712186 </s>
        1500  1507     -284407 </s>
Total score:    -30263429
```

Figure 3.14: Acoustic Score output file

Chapter 4

Conclusion

ASR is an important application nowadays it is being used in many fields such as telephony and knowing that there is a large number of Muslims around the world it became an interest to make such a great application to help the Muslims to learn and pronounce the Holy Quran in the correct way, so a research was done in understanding the ASR applications and the way they work and how to create and understand the Acoustic models and Language models and what is the theoretical background behind them. After finishing the research the actual work started by, first collecting the data of ten Quran readers and dividing the whole Quran text to 114 Sura and name all the files with a three bit counter in order to make calling them easier and in order to do that java scripts were made, second the Language model was created and researched more for more understanding of LM, third the Quran Aligner was used to align the Quran and make segmentation of each Sura, After that the data were prepared in order for them to get used in the Pocktsphinx application some figures explaining the files can be found in chapter 3 the implementation chapter, fourth after training and testing the results were taken Table 3.1 and we improved the aligning of the Quran aligner by using the new acoustic model that was generated after using Pocketsphinx and the same process was done agian but with the new AM. The last part of this project is the evaluating part, it is simple as the user can use only one command to test whether his/her reading were accurate or not from the Acoustic Score in the output from using the Sphinx3 aligner.

Chapter 5

Future Work

Due to the limited time the project was done on the basic level limited to only recognizing Quran speech with a Male's voice and evaluating only the correctness of the the whole segments. In the near future there will be improvements as we are planing to use not only Quran speeches for men only but we intend to use women and kids speeches in order to make the accuracy of the Am more higher and more efficient, also one of the futuristic planes it to make the evaluation for each phoneme and for each **Tashkel** marks in order to make the user pronounce in the most right and efficient way so we are planing to give each word and each phoneme a score different from the total score to test how much the reading are accurate.

Appendix

Appendix A

Lists

List of Figures

List of Tables

Bibliography

[1] Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition. Daniel Jurafsky James H. Martin.2ed edition, Draft of June 25, 2007.

[2] Huggins-Daines, David, et al. "Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices." Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on. Vol. 1. IEEE, 2006.

[3] CMU Sphinx 2016 ,http://cmusphinx.sourceforge.net/wiki/tutorialam.

[4] CMU Sphinx 2016 ,http://cmusphinx.sourceforge.net/wiki/pronunciation_evaluation.

[5] Osama A., ElMahdy M., Desouky A. Computer-Assisted Quran Pronunciation Learning System: Speech Recognition, Force Alignment, and Confidence Scoring, GUC, Cairo, Egypt ,2015.

[6] ElMahdy M., Hasegawa-Johnson M., Mustafawi E., Automatic Long Audio Alignment and Confidence Scoring for Conversational Arabic Speech, The 9th edition of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland, 2014.

[7] FFmpeg. A complete, cross-platform solution to record, convert and stream audio and video. https://www.ffmpeg.org/, 2015.

[8] P.R. Clarkson and R. Rosenfeld. Statistical Language Modeling Using the CMU-Cambridge Toolkit From Proceedings ESCA, Eurospeech, 1997.