Quran Tutor: An app to help Memorize the Quran

Raef Khan

Thomas Jefferson High School for Science and Technology

Abstract

There are millions of Muslims today who wish to memorize the Quran as part of their religion but are unable to do so due to multiple different factors. Speech recognition technology could aid in this task, by providing users with the flexibility needed in memorizing the Quran. Though there have been multiple research projects done on trying to create a speech recognition system for the Quran, which provide promising results, a Quran memorization aid application currently does not exist due to the complications and specifications needed to recite the Quran correctly, specifically the "Art of Tajweed." This project aims to continue developing the recognition model needed to correctly identify recitation of the Quran, using previous models and new technology developed for speech recognition, hoping to help those who wish to fulfill this endeavor.

Introduction

Islam is one of the fastest growing religions in the world today. One of the main aspirations of Muslims is to memorize the holy Quran. However, to do so usually requires a Hafiz, or someone who has perfected the memorization of the Quran, to help teach and review the Quran. In many parts of the world today, these Hafiz are difficult to find, while in other areas the student may not be able to pay the Hafiz, and for others there may be time constraints in which they are unable to keep a consistent review session with the Hafiz. As such, multiple virtual applications for reading and listening to the Quran have been created. Yet, nothing currently exists to help with reciting the Quran. This can be alleviated with the use of speech recognition technology.

Speech recognition is an exponentially growing phenomenon. With users continuously multitasking and trying to improve personal productivity, speech technology has become a basic necessity. For instance, people may use speech technology in cars for directions, at home to control appliances or perform menial tasks such as reminders and shop online, and in more advanced fields such as health care for notes and reports. Speech technology has also become more advanced in understanding multiple languages. This advancement of technology is incredible considering the amount of dialects and different tones a language may possess.

A good example of a language with multiple dialects is Arabic, where each country in the Middle East as well as different regions in each country has their own dialect for Arabic. To circumvent this, most speech recognition technology is done in Modern Standard Arabic (MSA). Most major speech recognition technology applications take hundreds of words from written data such as textbooks, newspapers, online articles,

etc. and create a speech recognition model from it. It will then break these words down into their phonetic sound, and then train for those sounds through the use of hundreds of audio files. Through the use of MSA as the default for speech technology using Arabic, mobile program producers use Arabic speech recognition in their application, and most people who speak and understand Arabic will be able to interact with the application.

However, an MSA recognition model does not work for the Quran. The Quran is read and recited differently than MSA and uses many words that are no longer common in current day Arabic. The main issue with the Quran is the "Art of Tajweed," which can be summarized as necessary elongation of vowels for 6 seconds, obligatory elongation of vowels for 4 or 5 seconds, optional elongation of vowels for 2, 4, or 6 seconds, nasalization of certain letters for 2 seconds, emphasized R, and unannounced letters. Different recitation techniques also create difficulty in producing a standard model for the Quran. For this reason, a special langage model for the Quran must be created. Afterwards, any application using such the Quran model must be completely accurate for the reciter wishing to review their recitation and continue memorization, as no mistakes are allowed in the recitation of the Quran. This research project aims to help continue building on other projects that try to create an efficient enough application to help Muslims memorize the Quran.

Background

Speech is the sound created when trying to communicate ideas using language as the delivery method. In its simplest component, each language has distinct sounds that when joined together create words and sentences. These sounds are known as phonemes.

There are about 40 distinct phonemes in English, while in Arabic there are approximately 30 consonant phonemes and six vowel phonemes.

Speech recognition is the ability of a machine to connect the distinct phonemes in a meaningful way to understand speech. The three main processes of speech recognition includes speech acquisition, feature extraction, and classification, as shown in figure 1 (Mengusoglu, 2004). Speech acquisition is the ability of the machine to convert the acoustic input signal into digital codes that can then be used by the machine. This is performed by an analog-to-digital converter, which takes specific measurements of the wave of the acoustic input at specific intervals, filtering out noise, and separating the digitized sounds based on frequency (Grabianowski, 2017).

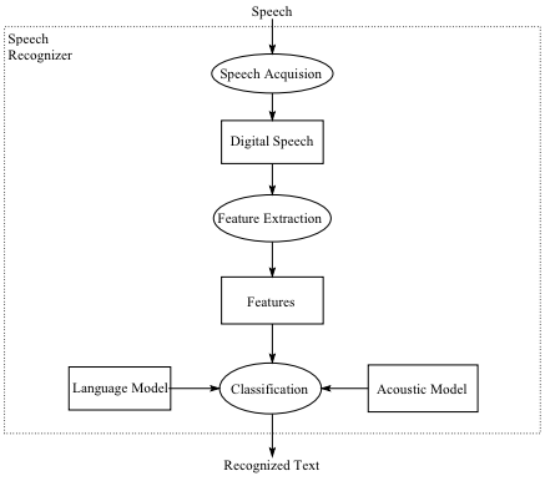

Figure 1: Speech Recognition Process

Speech is assumed to be created by a non-stationary random process. However, it is considered stationary for small windows of ten to thirty milliseconds (Gabor, 1947). Feature extraction is the ability of a system to extract the characteristics of these small portions of speech. A common method for feature extraction is the Mel-Frequency Cepstral Coefficient (MFCC) (Mengusoglu, 2004). MFCC is calculated by taking a frame of the speech, applying a windowing function, the most common for speech recognition being the Hamming window, taking the Fast Fourier Transformation (FFT) of the window, filtering the frequencies obtained by the mel-scale filter, taking the log of the

resulting frequencies, and then applying a Discrete Cosine Transformation (DCT), as shown in figure 2. The resulting coefficients are the vector representation for the frame chosen (Lyons, 2012).

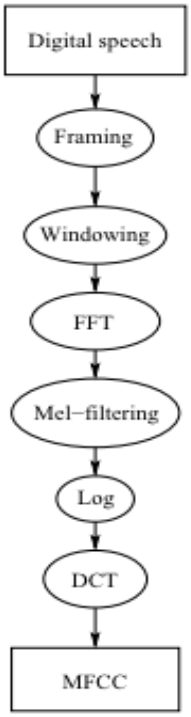

Figure 2: MFCC Process

Finally, the machine must create a pattern classification for the output of the feature extraction. Two main models, the acoustic model and language model, are used for classification. Acoustic models are the statistical representation of each class in speech recognition, which could be phonemes, sub-words, or words. Acoustic modeling is usually trained on large acoustic databases, which contain most word occurrences in a language (Gales, 2009). Most acoustic models use Hidden Markov Models (HMM) in their modeling today, however some have begun to blend HMM and Artificial Neural Networks (ANN) for better accuracy. Language modeling builds upon the acoustic by providing the natural probability of the next phrase, limiting the amount of searching needed (Mengusoglu, 2004).

One of the most commonly used toolkits for speech recognition is Carnegie Mellon's Sphinx IV. Used by both researchers and developers, Sphinx IV provides language models, acoustic training tools, a library of sources for multiple languages, the ability to be used for both mobile and web applications, and to be written in C and Java (Walker et al., 2004). Many researchers writing about the creation and usage of a Quranic recognition model use Sphinx IV to do so. In a study done by Amrani et al., using Carnegie Mellon's Sphinx IV and the MSA acoustic model provided less than 1.5% of

word error rate (WER) when all training and testing files were used for training. However, when unseen data was used, the WER spiked to about 50%. As such, Hassan et al. created an automatic delimiter for the Quran using Sphinx IV. This application used HMM (Hidden Markov Model), which allowed for the development of the acoustic model used. The program used one hour of training files on only one chapter of the Quran, Chapter 114, which is about 1/3 of a page long, and then tested it against reciters both with Tajweed accuracy and without. The results were promising, with about 90% accuracy achieved with the application. This application has been cited in many other research papers as well as proof that a recognition model for the Quran with high accuracy can be created as well.

The problem with creating a full Quran speech recognition is the amount of data and time necessary to train an accurate acoustic model. Sphinx requires multiple hours of training data from different speakers, and an ample amount of time ranging from a few days to months depending on the data to train the model. A problem with using the delimiter was the prerequisite of knowledge in language structure needed to code Sphinx IV, and the time it would take to train the data and test it in the time constraints of the project. Another issue with the delimiter is that it required reciters to already know the language and be able to recite correctly. As such, another research paper used a proof-of-concept idea named E-Hafiz to prove that model for the Quran can be used for those who are trying to learn it as well (Waqar et al., 2012). The E-Hafiz used Mel Frequency Cepstral Coefficient (MFCC) transformation to extract voice features to be used for more processing, and then Vector Quantization (VQ) for the acoustic modeling. Vector Quantization is the data clustering of the different vector outputs of the MFCC, and then

modeling probability density functions by the distribution of the vectors (Waqar et al., 2012). It combined vectors with similar values into clusters and represented them by their average value. This value is then split into 32 different means, creating 32 different clusters. In each cluster is a mean value named the code vector and all code vectors for each cluster saved is called the codebook. In terms of the user, the program uses the same method and then calculates the difference in the averages between the trained codebook and the users codebook. It is then compared to a threshold, which Waqar et al. set to 2.6 for beginners. At least three matches must be made for an utterance to be considered correct, otherwise it would be displayed as a mistake. The easiest way to code and use MFCC transformations and VQ was on MATLAB, which allows for the complex mathematical computations needed to produce an MFCC. MATLAB also allows for the easy change between live and audio file inputs, as well as conversions to C to implement into web applications.

The drawback to the E-Hafiz system was the minimalist amount of training done to proof the concept in using the application. Waqar et al. used a single training file, which contained a single word and then tested it against a single user speaking the same word, which it returned as an accurate match. However, the application does check for similarities in MFCC transformations, which is useful in finding varying frequency and pitch.

Developments

In order for the application to be able to recognize the recitation of the user and display his or her mistakes, the application would need both a back-end Quran speech recognition system and a front-end design simple enough for all types of users. For the

front-end, the design would first ask the user as to which verse they would like to recite. The application would then start recording the voice of the user reciting, without showing the verse. Once the user finishes his or her recitation, they will click on a “Done” button, which will send the recitation to a server that will test it using the speech recognition system specifically designed for the Quran.

This system needs to have a recognition model for the entire Quran, the best of which would be hard-coded as it is known each time what the next word is when reading specific verses, as well as hundreds of hours of training data to gain as much variance in recitation as possible. However, this would be an immense amount of data, as well as inefficient due to the amount of data written to correctly identify recitation. As such, a more varied recognition model must be created for the Quran. Since this level of specificity does not yet exist, a simpler method is the use of MFCC transformations and checking the difference between the reciter and the training files by Vector Quantization. As such, this application built on the E-Hafiz proof-of-concept program. It was expanded to take entire verses that range between four seconds and thirty seconds in length and used two training sets, recorded by expert reciters from Mecca, Islam’s holiest city, for each verse. The program then took the user’s recitation and compared the user’s MFCC transformations to the training files.

```
%-----------------------MFCC Train--------------------------------------

nc=12;
p=31;
n=128 ;
inc=floor(n/2);
fl=0.02;
fh=0.5;

%z=enframe1(s,hamming(n),inc);    % Overlaping of frames and apply Hamming Window
 [z,nf]=ovpframe(s,n,inc);

 win = hamming(n);
 w = win(:)';
 z = z .* w(ones(nf,1),:);

f=rfft(z.');                    % Fast Fourier Transformation

[m,a,b]=melbankm(p,n,fs);       % Mel-Frequency Filter Banks

pw=f(a:b,:).*conj(f(a:b,:));

pth=max(pw(:))*1E-6;
ath=sqrt(pth);                % Treshold

y=f(a:b,:);
y0=abs(y);
y1=m*y0;                       % Convertion to Mel Scale
y2=max(y1,ath);
y3=log(y2);
v=rdct(y3).';
nf=size(v,1);
nc=nc+1;
v(:,nc+1:end)=[];        %Consider Only 12 Coeff.
v(:,1)=[];               % Exclude zeroth Coeff.

%-----------------------------MFCC Train-------------------------
 code{chunkCnt} = kmeanlbg(v, k);    % VQ LBG Algo  (Construction of Codebook)
```

Figure 3: MFCC Training

Due to the original application using a single word as its training and testing file, the MFCC transformation occurred on the entire file, resulting in twelve “important” transformations, which are the first 12 mel transformations returned, that was then used for comparison. However, as more complexity was added to the program with files running longer than half a second, the program was edited to use a for-loop on the entire file, taking one-sixteenth of a second worth of data and producing an MFCC on that segment each time, which was then used for producing the codebook. Figure 3 shows the MFCC transformation being done on one of the frames of data. The files also had to be adjusted to the length of the training set, as to compare the same part of the audio with each other, rather than compare different utterances. Finally, to increase accuracy, the MFCC was run on overlapping windows by half the time. This means that the program did not instantly start at the next sixteenth of a second of audio, but rather went back and began at 1/32 of a second to 3/32 of a second, still moving at a 1/16 pace but not distinctly separating the parts of the audio file. Figure 4 shows the for-loop used to get the difference between the training and testing files for one specific frame, as well as the output for the average of all the frame differences.

```
% ------------------------MFCC Test----------------------

distmin = inf;
k1 = 0;

lc=length(code);                % No. Of Codewords In Code Book

for l = 1:lc                    % Check Each CodeBook
    d0 = disteusq(v, code{l});  %Find dist with corresponding codebook


    d1=min(d0,[],2);
    d2=sum(d1);
    d3=size(d0,1);
    dist = d2 / d3;         % Distorsion

    distance(1, 1) = dist;

    if dist < distmin
        distmin = dist;
        k1 = l;             % Speaker with Min. Distorsion
    end
end
r=r+1;
disp(distmin);
totaldist = totaldist+distmin;
count = count+1;
chunkCnt = chunkCnt + 1;
%disp(chunkCnt);
end
totaldist = totaldist/count;
if (totaldist < 2.6)
        givemsg = sprintf('Word Matched');
    else
        givemsg = sprintf('Word Not Matched');
end

    disp(givemsg);
    disp(totaldist);
```

Figure 4: Comparing training and testing data

In order to test the effectiveness of comparing MFCC transformations on specific frames of the audio file, I recorded multiple files with different speeds and purposeful

incorrect utterances. I then tested these files on the trained files of the two master reciters, graphing the resultant frames. When looking at the specific locations at which the incorrect speech occurred, the program was sometimes able to provide a slightly higher difference in MFCC transformations that that with a normal reading. However, the average difference for the entire file tended to be between 3.5 and 5 on the mel-scale, much higher than the 2.6 difference Waqar et. al had established. While the definitive reason for this is unknown, it is likely due to the MFCC producing a feature vector of the frequency of the frame rather than the actual phoneme spoken at the frame, which may be a different frequency than the trained data, but still be the same phoneme. There were other locations where a spike in the difference of the MFCC transformation occurred, though the spoken sound at that location was the same for the testing and training file.

Another issue is with the training files for the two reciters, as one had a mono channel while the other was a stereo channel. This proved to have different outputs in the feature extraction. Restricting the feature extraction to only the mono channel seemed to have a slight effect on the output, meaning that more needs to be done in researching audio files with mono and stereo channels.

Due to the insufficient amount of time, an accurate recognition model for the entire Quran was unable to be completed. The system would need hundreds of files of data from expert reciters, of which verse by verse file data is not easily found. The model would then have to produce its own acoustic and language model, in order to be robust and not hard coded for each verse.

The model must also be much more specific than most recognition models to be accepted by Hafiz and scholars, since the Quran does not allow for any mistakes in

recitation. These include mistakes in length a vowel was spoken, utterance of Tajweed, and phonetic guide sounds. For instance, a standard Arabic model might understand the word “Quittun” and “Quittan”, meaning cat, as the word cat. However, the ending of the word defines its grammatical structure, with an “un” ending as being the subject of a sentence while an “an” ending being the object of a sentence. This incorrect utterance is not allowed in the Quran, as it may change the meaning of a phrase. Therefore, the recognition system must be able to also accurately identify the phoneme spoken to build the phonetic guide on the words, rather than just understanding the sentence spoken. That way, it may more easily recognize mistakes in phonetic reading as well as Tajweed reading.

Due to the multiple inaccuracies displayed by the VQ method, the application was never tested with other users and a front-end design was never created for publishing. If a published app was created, however, it would ask the user to choose a verse, record them speaking the verse, and then display the verse and highlight segments that were misspoken. The highlighted segment would contain information about the mistake made, as well as an explanation as to what the actual recitation should be, based on whether it was a pronunciation error or Tajweed error. It would also include the audio of the user and the master to allow the user to listen to their mistake versus the accurate recitation.

Discussion

The E-Hafiz program was unable to recognize longer verses and unique recitation patterns, therefore unreliable as a source for helping users memorize the Quran. The VQ took the MFCC transformations and found their mean values based on clusters, which inadvertently meant that it was testing for frequency rather than the phonemes spoken.

However, an acoustic model would have converted the MFCC transformations back into phonemes, which is more useful in the case of memorizing the Quran to the preciseness needed. The conversion to phonemes would allow the application to better define what utterance was said rather than the correct utterance. In this way, the application ignores frequency of a person, and instead focuses on utterance. Using a language model for the application would also increase efficiency due to the limited number of cases of which phoneme would proceed the previous in a fixed set of data.

For example, a female reciter would likely have a higher frequency when reciting, however her recitation would still be correct as long as she adhered to the Tajweed rules of the Quran. In VQ, the codebook would cluster her different frequencies of recitations rather than the phonemes spoken, resulting in a mismatch on almost every level. However by using an acoustic model, her recitation would be shown as correct. In order to accompany the millions of people who would be using this app from all parts of the world with different dialects, the application must not take into account individual variances in recitation, but instead analyze the differences of the phonemes between the trained audio and the user's.

Through this research, it is apparent as to why a Quran application that uses speech recognition to help users memorize the Quran has not yet been created. The process requires immense amount of training and testing, and while there have been promising advancements in the field, being able to fully recognize the Quran is still a challenge. Though simple in theory, the complexities and specifications for reciting the Quran make it difficult to produce an application with enough accuracy to help those who wish to memorize it. However, with more research into the improvement of speech

recognition accuracy, as well as recording more data files for training the recognition model, this application may soon be a possibility.

References

Anusuya, M. A., & Katti, S. K. (2009). Speech recognition by machine: A review. *International Journal of Computer Science and Information Security*, *6*(3), 181-205. Retrieved from https://arxiv.org/pdf/1001.2267.pdf

El Amrani, M. Y., Rahmanb, M.M. H., Wahiddinb, M. R., & Shahb, A. (2016). Building CMU Sphinx language model for the Holy Quran using simplified Arabic phonemes. *Egyptian Informatics Journal*, *17*(3), 305-314. https://doi.org/10.1016/j.eij.2016.04.002

Figure 1. Speech Recognition Process. Reprinted from *Con dence measures for speech/speaker recognition and applications on Turkish LVCSR* (Doctoral dissertation), 13, by E. Mengusoglu, 2004. Retrieved from http://www.tcts.fpms.ac.be/publications/phds/mengusoglu/thesis_mengus.pdf

Figure 2. MFCC Process. Reprinted from *Con dence measures for speech/speaker recognition and applications on Turkish LVCSR* (Doctoral dissertation), 13, by E. Mengusoglu, 2004. Retrieved from http://www.tcts.fpms.ac.be/publications/phds/mengusoglu/thesis_mengus.pdf

Gales, M. (2009). *Acoustic modelling for speech recognition: Hidden markov models and beyond?* [PowerPoint slides]. Retrieved from http://mi.eng.cam.ac.uk/~mjfg/ASRU_talk09.pdf

Grabianowski, E. (2017). How speech recognition works. Retrieved from http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition1.htm

Lyons, J. (n.d.). Mel frequency cepstral coefficient (MFCC) tutorial. Retrieved from Practical Cryptography website: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

MathWorks, T. (2017). Hidden Markov Models (HMM). Retrieved from https://www.mathworks.com/help/stats/hidden-markov-models-hmm.html

Mengusoglu, E. (2004). *Con dence measures for speech/speaker recognition and applications on Turkish LVCSR* (Doctoral dissertation). Retrieved from http://www.tcts.fpms.ac.be/publications/phds/mengusoglu/thesis_mengus.pdf

Mohammed, A., & Bin Sunar, M. S. (2014). Verification of Quranic verses in audio files using speech recognition techniques. *1st International Conference of Recent Trends in Information and Communication Technologies*. Retrieved from http://www.ysrgst.org/irict/wp-content/uploads/2014/10/37.pdf

Muhammad, A., ul Qayyum, Z., Mirza, W. M., Tanveer, S., Martinez-Enriquez, A.M., & Syed, A. Z. (2012). E-Hafiz: Intelligent system to help muslims in recitation and memorization of Quran. *Life Science Journal*, *9*(1), 534-541. Retrieved from http://www.lifesciencesite.com/lsj/life0901/080_8204life0901_534_541.pdf

Muhammad, W. M., Muhammad, R., Muhammad, A., & Martinez-Enriquez, A. M. (2010). Voice content matching system for Quran readers. *Ninth Mexican International Conference on Artificial Intelligence*. Retrieved from https://www.researchgate.net/publication/224214181_Voice_Content_Matching_System_for_Quran_Readers

Razak, Z., Ibrahim, N. J., Bin Idris, M. Y. I., Tamil, E. M., Yakub, M., Yusof, Z. M., & Abdul Rahman, N. N. (2008). Quranic verse recitation recognition module for support in j-QAF learning: A review. *IJCSNS International Journal of Computer Science and Network Security*, *8*(8), 207-216. Retrieved from http://paper.ijcsns.org/07_book/200808/20080831.pdf

Soname, A. (n.d.). *Sheikh Mahir verse by verse Quran* [Audio file]. Retrieved from http://www.quranaudio.info/2013/02/sheikh-mahir-verse-by-verse-quran.html

Soname, A. (n.d.). *Sudais verse by verse Quran* [Audio file]. Retrieved from http://www.quranaudio.info/2011/02/sheikh-abdul-rahman-al-sudais.html

Tabbal, H., Al-Falou, W., & Monla, B. (2007). Analysis and Implementation of an Automated Delimiter of "Quranic" Verses in Audio Filesusing Speech Recognition Techniques. In M. Grimm & K. Kroschel (Eds.), *Robust speech recognition and understanding* (pp. 351-362). Retrieved from http://www.intechopen.com/books/robust_speech_recognition_and_understanding/analysis_and_implementation_of_an_automated_delimiter_of__quranic__verses_in_audio_files_using_speec

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., . . . Woelfel, J. (2004). Sphinx-4: A flexible open source framework for speech recognition. *Sun Microsystems*. Retrieved from http://delivery.acm.org/10.1145/1700000/1698193/smli_tr-2004-139.pdf?ip=198.38.31.9&id=1698193&acc=NO%20RULES&key=53834514706649C1%2E53834514706649C1%2E4D4702B0C3E38B35%2E4D4702B0C3E38

B35&CFID=766857577&CFTOKEN=41849163&__acm__=1495817921_47df1f43b8f9644718ec41a6ff470030

Yekache, Y., Mekelleche, Y., & Kouninef, B. (2011). Towards Quranic reader controlled by speech. *International Journal of Advanced Computer Science and Applications*, *2*(11), 134-137. Retrieved from https://thesai.org/Downloads/Volume2No11/Paper%2023-%20Towards%20Quranic%20reader%20controlled%20by%20speech.pdf

Zainon, N.S. Z., Ahmad, Z. A., Romli, M. A., & Yaacob, S. (2012). Speech quality based on Arabic pronunciation using MFCC and LDA: Investigating the emphatic consonants. *2012 IEEE International Conference on Control System*. https://doi.org/10.1109/iccsce.2012.6487178