DCU Engineering & Computing Assignment Submission

Assignment: Speaker Recognition (CA641**)**

Karl O hInneirghe (12212130), John A Ryan (12211911)

MSSF1 - MSc in Security & Forensic Computing

karl.ohinneirghe3@mail.dcu.ie , john.ryan79@mail.dcu.ie

Abstract

This fixed utterance verification system uses the HTK tool by Cambridge University Engineering Department. It provides a one-to-one comparison of a fixed phrase by a "Speaker", whom we wish to verify, and a set of "Imposters" whom we wish to reject. This was achieved by building a client profile from the Mel Frequency Cepstral Coefficient vectors extracted from the Speaker sample and using these to train the system using Gaussian Mixture Models. The imposter's data was then run through the same model to extract a set of log likelihood's from which to determine an Equal Error Rate that we used as out Accept/Reject marker. A series of comparisons were then used to analyze the success of our experiment.

1. Introduction

Automatic Speech Recognition Systems (ASRS) are systems that take the utterances or speech from humans as an input and converts it to a machine understandable format to form a common connection with spoken data. According to Markowitz [1, p. 21] a speech recognition system when it functions correctly has three primary tasks.

1. Preprocessing
2. Recognition
3. Communication (Duplex)

Preprocessing converts the spoken input into a form the recognizer can process. The recognition step identifies what has been said and the communication step will send the processed input to the software/hardware system. These three primary tasks remain invisible to the end user using the system but indirectly they would notice them if the speed or accuracy was affected. Becchetti [2, p. 6] says that in human ASR systems, humans rely on a parametric model where knowledge can be gained by parameter tuning. These parameters are set so the model can recognize phenomena in the most accurate way. The larger the number of repetitions, the more accurately the model reflects real examples of speech. This stage is referred to as "training" e.g. Babies are able to recognize a consistent number of words after about "two years of training".

Although human's process information using large neural networks, the ASR uses the simplest Hidden Markov Models (HMMs), however, in this experiment we will be using Gaussian Mixture Models (GMMs). In order to perform speech recognition, the ASRS processes the sequence of samples extracted from recordings. The input is thought of as linguistic units. During this phase the signal (samples) go through different steps: pre-emphasis, frame-blocking, windowing, feature extraction and MFCC.

For this assignment the class decided collectively to use the same phrase to train and test an ASRS. We chose "A boring novel is a superb sleeping pill" based on a phonetically rich sentence that was suggested by the linguistic data consortium hosted by the University of Pennsylvania [3]. The system was trained with a single speaker with nine samples to be used as imposters to identify how close or where the point of equality is against the trained and test data set. This paper will implement an ASRS using Gaussian

Markov Models and demonstrate three comparative experiments to investigate its performance.

2. Speaker Verification System Implementation

To demonstrate our implementation of a speech verification system we used the Hidden Markov Model Toolkit (HTK) [4] on Ubuntu [5] and used GNU Octave [6] for numerical calculations and graphing. We used a single speaker uttering a single phrase and compared the output generated by our system to that of eight other speakers whose samples were mixed together in various configurations. All code, commands and scripts used in this section can be found in the Appendix A.

2.1 Data Preparation

The set of voice samples used was collected from nine class members and made available to all. The specifics of which can be seen in Table-2.1 below. Before performing analysis of any of the recordings, the system setup was configured and the recordings prepared. We followed the instructions in the tutorial provided [7] with some notable exceptions that would allow us to perform fixed utterance verification using Gaussian Mixture Models.
First, we created a single speaker grammar and word network using HTK's HParse Tool. Next, to allow for HTK to process the utterances we parameterized the recordings, that is converted them to a series of acoustical vectors using the Mel Frequency Cepstral Coefficient (MFCC) technique using the HCopy tool. This extracted the features of the recordings that we wanted to focus our system on. The configuration elements of note can be seen in Table-2.1. The Source Rate (Sampling Period) and the WINDOWSIZE (Frame duration) were agreed upon by the class and Lecturer in an interactive session as being well proven settings for the system. With the system configured each recording was run through the feature extraction and ".mfc" files created containing the vectors of the recording to be analysed.

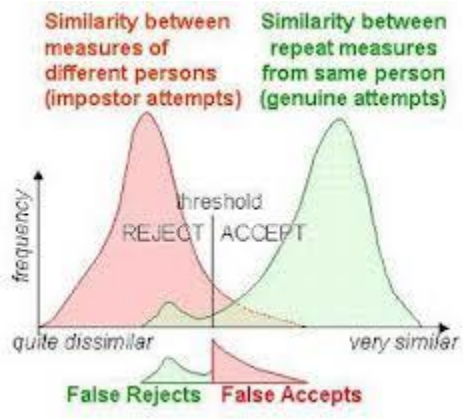
Table-2.1: Recording Guidelines and HTK Configuration

Utterance Guidelines	**Key HTK Configuration Elements**
# Samples: 10 Sample Rate: 16000Hz Bit Rate: 16bit Channels: Mono Phrase: "A boring novel is a superb sleeping pill."	TARGETRATE=100000 WINDOWSIZE=300000 NUMCEPS=12

2.2 Model Preparation and Data Estimation

To configure the system to use GMMs we must create a prototype file that will be used during the training phase for estimating the optimal parameters (transition probability, mean and variance vectors for each observation function [8, p. 362]. Instead of the five states used during the HMM model setup, we will use 3 states (including two null states) which is all that is required for the Gaussian technique. The probability matrix is also changed to a 3x3 matrix with no reverse paths allowed. To get meaningful data from which to draw a comparison, we must train our system with the Speakers data. Once trained, using the HCompV tool, we then had an iterative process where we re-estimated the model using the HERest tool, in this case three times to reduce the convergence factor of the data under test [8, p. 362].

Figure-2.1: Setting the threshold [9]



2.3 Data Analysis

The final step in the HTK process was to use the HVite tool to generate a set of Log Likelihood scores from the re-estimated Gaussian Mixture Models that were generated in the previous step. We do this first for our Speaker model data and then again for our imposter data. Running the imposter data through the Speakers estimation allows us to create Log Likelihood's to compare the two models and compute the normal probability density functions of the

speaker and the imposters. Once plotted, the observed intersection of the two curves provided us with the means to calculate the "equal error rate", that is, the equal areas under the curve that allow us to define our False Acceptance/ False Rejection Rate Threshold.

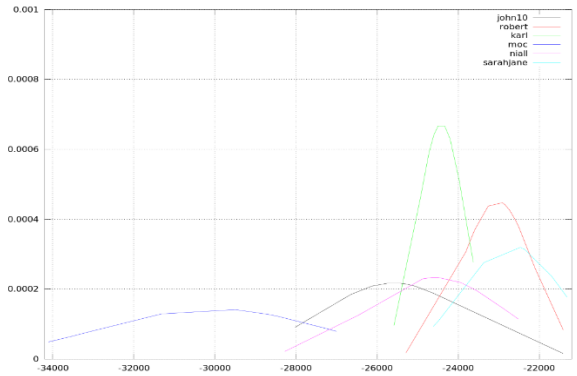
2.3 Comparative Tests

For some experimentation we have taken alternative ways to test the system. The initial test is the speaker saying the sample phrase. We will also have the speaker saying a different phrase. And finally we will look at what happened when we increased the sample set of imposters and what we think it means.

3. Results Analysis

To analyze the performance of the system, we took six of the nine speakers to represent a sample space. The distribution of their log likelihoods can be seen in the graph below.

Figure-4.1: Distributions of all samples involved



3.1 Threshold Analysis

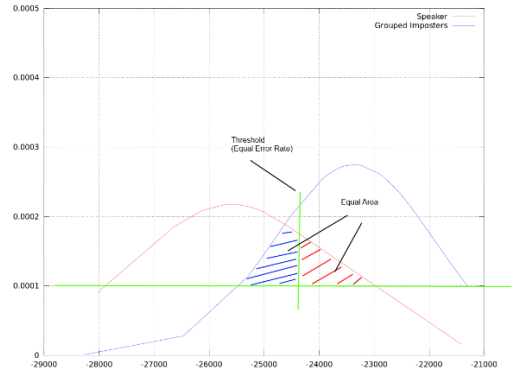
The system was tested for a "Speaker" and its performance against various imposter combinations. To allow us to estimate a tolerance for the system in determining the accuracy of accepting/rejecting samples, we compared the Normal probability distribution curves for each of the Speaker, a group of imposters made up of 3 speakers and finally a mix of all speakers.

Table-3.1: Results

	# Voices	# Samples	Mean	STD
Speaker	1	10	-25597	1833.60
Group Imposter	3	35	-23364	1451.70
All Imposters	6	65	-24767	3016.50

From the graph below we can see that the intersection of the two distributions gives us a distinct overlap and allows us a clear "brute force" means to determine the Equal Error Rate.

Figure-3.2: Speaker vs Grouped Imposters



After taking a series of arbitrary points along the x-axis and calculating the area under the curve limited by those threshold values, we determined an equal error rate of 24375. Meaning that any model that we estimate that falls less than the threshold value will be accepted as a positive. Even from eyeballing the graph above, we can see that a large number of Imposters results fall to the left of that line. These are considered False Acceptance. Conversely, any of the speaker's data that falls to the right of the threshold are falsely rejected.

3.2 Data Comparison

To further examine the system above, we have taken a look at three practical cases to compare the results to. The first is the main speaker the same test phrase to see where in the accept/reject domain they fall.

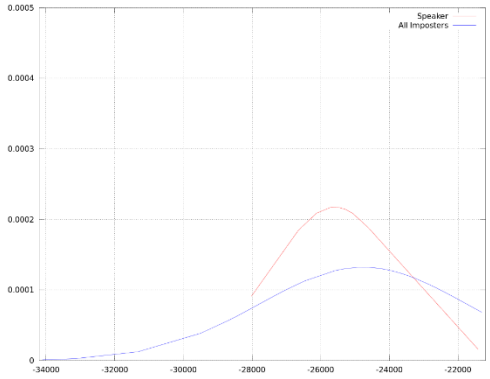
Table-3.2: Comparison results

Test	Log Likelihood	Result
Speaker saying same phrase	-26122	Speaker correctly accepted
Speaker saying different phrase	-24152	Speaker correctly rejected

The second is the speaker saying the phrase “There is nothing good on TV”, a phrase similar in length to the original utterance. See Appendix D.

The final experiment was to create a broader imposter sample space from which to draw a more refined Equal Error Rate. With the seemingly high crossover it was decided to increase the sample space of our imposters to refine the threshold. This however did not give better results as the distributions converged making it quite impossible to draw a more accurate Equal Error Rate as can be seen form Figure 3.3

Figure-3.3: Speaker vs. All Imposters



4. Conclusion

Our experiment has shown that we can create an Automatic Speech Recognition system using Gaussian Mixture Models for fixed utterance speaker verification, and that this system can be derived from a speaker recognition system using Hidden Markov Models. The results of the experiment do show that further research is required to refine the thresholds that are defined for Acceptance/Rejection, this would include increasing the sample size to perform system testing to get a figure for the False Acceptance Rate and False Rejection Rate. There is also room to tweak the quantization parameters to see if different windowing and framing will produce better MFCC inputs. Finally it would be of interest to repeat the experiment using HMMs to see the affect another model would have on the outputs.

5. Acknowledgments

We would like to thank our class for their interactions on Moodle and all those who posted their recordings to the public share.

6. References

[1] J. Markowitz, "Using Speech Recognition," 1998.

[2] Becchetti, "Speech Recognition," Wiley, 1999, pp. 6, 125.

[3] L. D. Consortium, "CSLU's Phonetically Rich Phrases," University of Pennsylvania, 2010. [Online]. Available: http://www.ldc.upenn.edu/Catalog/docs/LDC2008S09/manual/html/node16.html. [Accessed 28 04 2013].

[4] HTK, Cambridge University Engineering Department (CUED), [Online]. Available: http://htk.eng.cam.ac.uk. [Accessed 16 04 2013].

[5] Canonical Ltd, "Ubuntu 12.04 LTS," Canonical Ltd, [Online]. Available: http://www.ubuntu.com/. [Accessed 16 04 2013].

[6] GNU Octave, "http://www.gnu.org/software/octave/," Open Source, [Online]. Available: http://www.gnu.org/software/octave/. [Accessed 16 40 2013].

[7] J. McKenna, "Building a Simple Identification System," DCU, 2012. [Online].

[8] R. V. K. D. Mohit Dua1, "Punjabi Automatic Speech Recognition Using HTK," july 2012. [Online].

[9] F. P. Fandos, "How to shade region under a function in octave," Stackoverflow, 2012. [Online]. Available: http://stackoverflow.com/questions/13445168. [Accessed 20 04 2013].

[10] S. Albayrak, "http://www.ce.yildiz.edu.tr/personal/songul," [Online].

[11] K.-c. Lee, "http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html," [Online].

[12] J. Holmes, "Speech Synthesis and recognition," 1988.

7 Appendix

Appendix A

HTK Commands

```
# Crate a word network
HParse gram.txt wdnet
HCopy -T 1 -C config_wav2mfc -S convert.scp
HList Data/ma1.mfc

# Train the Model
mkdir hmm0
HCompV -C config_mfc -f 0.01 -m -S training.scp -M hmm0 proto
mkdir hmm1
HERest -C config_mfc -I speakertrainmodels0.mlf -S training.scp -H hmm0/macros -H
hmm0/hmmdefs -M hmm1 phonemodels0
mkdir hmm2
HERest -C config_mfc -I speakertrainmodels0.mlf -S training.scp -H hmm1/macros -H hmm1/hmmdefs
-M hmm2 phonemodels0
```

```
mkdir hmm3
HERest -C config_mfc -I speakertrainmodels0.mlf -S training.scp -H hmm2/macros -H hmm2/hmmdefs
-M hmm3 phonemodels0

# All Results for speaker
HVite -H hmm3/macros -H hmm3/hmmdefs -S training.scp -i results_speaker.mlf -w wdnet dict
HmmList
HResults -I speakertrainmodels0.mlf HmmList results_speaker.mlf

# Imposters
HVite -H hmm3/macros -H hmm3/hmmdefs -S ImposterTestAll.scp -i results_imposter.mlf -w wdnet
dict HmmList
HResults -I ImposterTestmodelAll.mlf HmmList results_imposter.mlf
```

Appendix B

Octave commands

```
figure(1)
clf
hold
grid on
#legend("Speaker against 3 person imposter sample set")
plot(john10, normpdf(john10, mean(john10), std(john10)), ";john10;1")
plot(sarahjane_robert_naill, normpdf(sarahjane_robert_naill,
mean(sarahjane_robert_naill), std(sarahjane_robert_naill)), ";sarahjanerobertnaill;3")
print('plotSpeakerVs3Samples.png')
figure(2)
clf
hold
grid on
#legend("Speaker against 5 person imposter sample set")
plot(john10, normpdf(john10, mean(john10), std(john10)), ";john10;1")
plot(imposter, normpdf(imposter, mean(imposter), std(imposter)), ";All Imposters;3")
print('plotSpeakerVs5Samples.png');
figure(3)
clf
hold
grid on
#legend("All Samples in Set")
plot(john10, normpdf(john10, mean(john10), std(john10)), ";john10;0")
plot(rcollins, normpdf(rcollins, mean(rcollins), std(rcollins)), ";robert;1")
plot(karl, normpdf(karl, mean(karl), std(karl)), ";karl;2")
plot(moc, normpdf(moc, mean(moc), std(moc)), ";moc;3")
plot(nrooney, normpdf(nrooney, mean(nrooney), std(nrooney)), ";niall;4")
plot(sarahjane, normpdf(sarahjane, mean(sarahjane), std(sarahjane)), ";sarahjane;5")
print('plotAllSamples.png');
```

Appendix C

Speaker Data Log Likelihood Values

```
# name: john10
-28015.45703 -27638.80273 -26649.98438 -26122.86328 -25706.50781 -25477.16211 -
25275.28125 -25060.60156 -24610.11523 -21417.99414
```

```
# name: rcollins
-25289.44531 -23807.58203 -23719.82031 -23591.69141 -23271.74219 -23250.19336 -
22908.79102 -22899.07617 -22843.89844 -22732.67383 -22577.87109 -22571.80469 -22501.61719
-22102.47461 -21410.45703
# name: karl
-25582.37891 -24928.67773 -24721.79492 -24599.60938 -24494.51953 -24332.63477 -
24197.64453 -23977.28125 -23684.36133 -23625.60547
# name: moc
-34100.57422 -33569.92188 -33130.44531 -31317.06445 -29495.83203 -28694.0625 -28466.32227
-27300.66406 -27222.45898 -27009.47852
# name: nrooney
-28269.48828 -26469.47461 -24875.17773 -24658.99609 -24492.64258 -23953.05664 -
23901.13867 -23544.77539 -22885.56641 -22523.13477
# name: sarahjane
-24612.16016 -24483.86914 -23371.21875 -23367.46875 -22467.3418 -22356.14258 -21699.56055
-21508.97656 -21502.22852 -21314.29688
# name: sarahjane_robert
-25289.44531 -24612.16016 -24483.86914 -23807.58203 -23719.82031 -23591.69141 -
23371.21875 -23367.46875 -23271.74219 -23250.19336 -22908.79102 -22899.07617 -22843.89844
-22732.67383 -22577.87109 -22571.80469 -22501.61719 -22467.3418 -22356.14258 -22102.47461
-21699.56055 -21508.97656 -21502.22852 -21410.45703 -21314.29688
# name: sarahjane_robert_naill
-28269.48828 -26469.47461 -25289.44531 -24875.17773 -24658.99609 -24612.16016 -
24492.64258 -24483.86914 -23953.05664 -23901.13867 -23807.58203 -23719.82031 -23591.69141
-23544.77539 -23371.21875 -23367.46875 -23271.74219 -23250.19336 -22908.79102 -
22899.07617 -22885.56641 -22843.89844 -22732.67383 -22577.87109 -22571.80469 -22523.13477
-22501.61719 -22467.3418 -22356.14258 -22102.47461 -21699.56055 -21508.97656 -21502.22852
-21410.45703 -21314.29688
# name: imposter
-34100.57422 -33569.92188 -33130.44531 -31317.06445 -29495.83203 -28694.0625 -28466.32227
-28269.48828 -27300.66406 -27222.45898 -27009.47852 -26469.47461 -25582.37891 -
25289.44531 -24928.67773 -24875.17773 -24721.79492 -24658.99609 -24612.16016 -24599.60938
-24494.51953 -24492.64258 -24483.86914 -24332.63477 -24197.64453 -23977.28125 -
23953.05664 -23901.13867 -23807.58203 -23719.82031 -23684.36133 -23625.60547 -23591.69141
-23544.77539 -23371.21875 -23367.46875 -23271.74219 -23250.19336 -22908.79102 -
22899.07617 -22885.56641 -22843.89844 -22732.67383 -22577.87109 -22571.80469 -22523.13477
-22501.61719 -22467.3418 -22356.14258 -22102.47461 -21699.56055 -21508.97656 -21502.22852
-21410.45703 -21314.29688
```

Appendix D

Compare Data HTK Commands

```
# Phrase: "There is nothing good on TV"
HVite -H hmm3/macros -H hmm3/hmmdefs -S ImposterSpeakerCompare.scp -i results_compare1.mlf -w
wdnet dict HmmList
"Data/jryancompare01.rec"
0 28300000 Speaker -24152.822266
```

Appendix E

Sample Data Location

```
All recordings are in a shared location on DropBox administered by John Ryan
https://www.dropbox.com/l/tqrcBSQNEqsIl7QR7h5OIe/invite
```