Automatic Speech Recognition using HTK

William L. and Andrea T.
Data Mining and Machine Learning

Recognition Result with Static MFCC Features

Executing the Perl script that trains and tests the given data gives us a result that is outputted into the *result* directory under the name *recognitionFinalResult.res*. The recognition results of the clean tests using a system of only clean training data with static features are shown in Fig. 1.

```
---------------------- Overall Results --------------------------
SENT: %Correct=63.54 [H=122, S=70, N=192]
WORD: %Corr=92.02, Acc=88.67 [H=796, D=9, S=60, I=29, N=865]
================================================================
```

Fig. 1. *Recognition result without extra features using clean data. No changes in -p flag value.*

From Fig. 1, the results on sentence (SENT) and word (WORD) level are very different. These values indicate how well the entire words and sentences are recognised, so it can be deduced that this system is good at recognizing words, but bad at recognizing sentences, at a low accuracy of 63.54%. Additionally, at the word-level result, there is an imbalance between the terms *D* (deletion) and *I* (insertion). That can be changed by modifying *-p* flag value when executing the *HVite* command.

After testing, it was found out that a *–p* value of -15 gave the most balance between *D* and *I* as seen in Fig. 2. It was also discovered that by increasing the *-p* flag, *I* increase, and further increasing *-p* would also increase the difference between *D* and *I*. As expected, decreasing *-p* increases *D* and decreases *I*, allowing us to find a value where *D* and *I* are identical. The balance of *D* and *I* has also increased the correctness at the sentence level and the accuracy at the word level.

```
---------------------- Overall Results --------------------------
SENT: %Correct=68.75 [H=132, S=60, N=192]
WORD: %Corr=91.68, Acc=90.17 [H=793, D=13, S=59, I=13, N=865]
================================================================
```

Fig. 2*. Recognition result without extra features using clean data. -p = -15.*

Delta and Delta-Delta Features

Similarly, we can include the Δ and $\boldsymbol{\Delta^2}$ features into the recognition system using the same script with some modifications. Values in the Perl script containing `MFCC_E` are changed into `MFCC_E_D_A`. Δ and $\boldsymbol{\Delta^2}$ features have a total of 39 dimensions, three times more than the initially 13 dimensions of the MFCC_E feature. The last modification in the Perl Script is what prototype to use for the HMM training. The prototype file is located inside the *lib* directory, and the modification to the prototype looks like `$Proto = "$REC_DIR/lib/proto_s1d39_st8m1_LabDMML_MFCC_E_D_A";`, where `$REC_DIR` is the directory of the project.

Other than Perl scripts, some config files inside the config directory will also need modifications, mainly the files: *config_HCopy_MFCC_E*, *config_test_MFCC_E*, and *config_train_MFCC_E*. Values for `TARGETKIND` in all three files needs to be modified to `MFCC_E_D_A` and in the *config_HCopy_MFCC_E* file, set the `DELTAWINDOW = 3` and `ACCWINDOW = 2`. After all the modifications in the Perl script and config files, as well as trying different *-p* values, the Perl script was executed using clean training and test data. As seen in Fig. 3, the result is outputted again into *recognitionFinalResult.res*, unless the output file name was changed in the Perl script.

```
---------------------- Overall Results -------------------------
SENT: %Correct=88.02 [H=169, S=23, N=192]
WORD: %Corr=97.57, Acc=96.88 [H=844, D=6, S=15, I=6, N=865]
================================================================
```

Fig. 3. *Recognition result including Δ and* $\boldsymbol{\Delta^2}$ *features using clean data. -p = -50.*

Fig. 3 has shown a significant increase in correctness and accuracy at both levels. The *D* and *I* values are balanced, showing maximum accuracy for the recognition system when testing clean data. Correctness at the sentence level has jumped up by around 20%, and accuracy at the word level is at 96.88% with only 21 missed words out of 865, while using Δ and $\boldsymbol{\Delta^2}$ features.

With Δ and $\boldsymbol{\Delta^2}$ features, the recognition system was able to recognize most of the testing data, given that it now has three times the features than before. The increase in accuracy was expected because the Δ and $\boldsymbol{\Delta^2}$ features approximate the first and second derivatives of the signal. By deriving the signal, we get an approximation of the signal energy based of off its previous energy. If a signal energy was high previously and is now low, we would have a negative gradient, and a similar process happens to get a positive gradient. Compared to the system without Δ and $\boldsymbol{\Delta^2}$ features, static systems would not be able to recognize if an energy signal came from a high or low energy, while this new

system can. With this, the system now has more input data and parameters that it could learn from, which may or may not lead to an increase in accuracy. But in this case, the accuracy did increase and that means that a pattern was found in the approximation of the signals that the system could learn and recognize.

3 Gaussian Mixture Components

Previous recognition systems have only been using one Gaussian per state, so the following system will employ three Gaussian mixture components per state, and the results will be compared with previous obtained results.

Modifications will be needed in the Perl script. Firstly, we will need to include the two additional mixtures that are in the *lib* directory named *mix2_st8_LabDMML.hed* and *mix3_st8_LabDMML.hed* into the list of global variables. Secondly, two additional Gaussians per mixture will need to be created and iterated after training the 'stop pause (SP)' model. The process is like when creating the SP model using *HHEd* to increase the mixture and iterate using *HERest*. After creating the model with three Gaussians per mixture, we can run an additional six iteration on the model to train it. While trying out different *-p* values, the Perl script was executed using clean data and the results are compared to the previous system with a single Gaussian mixture in Table 1. More details on the following results (*H, D, I, S,* and *N* values that make up results) can be seen in Appendix 1.

Table 1*: Recognition result comparison for single and three gaussian mixture per state, trained and tested using clean data*

	%Correctness		%Accuracy	
	SENT	WORD	WORD	*-p*
1 Gaussian Mixture	88.02	97.57	96.88	-50
3 Gaussian Mixtures	91.67	98.38	98.03	-40

From Table 1, it can be seen that by employing a three Gaussian mixture components per state, the correctness and accuracy percentage have increased, albeit by a small margin. With a large dataset, assuming it has multiple speakers, a single Gaussian system would be inadequate to model the variation in speech patterns correctly. Though it may get close to modelling it correctly, there will still be small errors and these are solved by having multiple Gaussian mixtures, in this case, three.

Effect of Noise

In this part, we explore the effect of noise by testing two different decibels (SNR10 and SNR20) of noisy data using the previously built three gaussian model that was trained using clean data. For this, we can use the *onlyTest* Perl script and modify some of the global variables to test the noisy data using

a built model. Firstly, for `$LIST_TEST` and `$LIST_Test_HCopy`, we need to change which list needs to be called. Instead of `CLEAN1`, it was modified to `N1_SNR10` and `N1_SNR20`, for test one and test two, respectively. Secondly, our already built three Gaussian model was saved in the *hmmsTrained* directory, and iterated until *hmm20*, so the *models* and *macro* files would be inside *hmm20* and not in *hmm8*. While trying different *-p* values for a balanced *D* and *I* value, we executed the *onlyTest* Perl script for the two different noisy sets of data. The recognition results for testing two different noisy sets of data are presented in Table 2. More details on the following results can be seen in Appendix 2.

Table 2*: Recognition result comparing SNR10 and SNR20 using the previous system with three Gaussian model*

	%Correctness		%Accuracy	
	SENT	WORD	WORD	*-p*
SNR10	13.02	67.28	59.08	-86.5
SNR20	68.23	93.06	90.98	-57.5

From Table 2, training with only clean data does not provide reliable results when testing with noisy data, especially at the sentence level. The correctness when testing SNR10 is dreadful, at a low value of 13.02%. The low inaccuracy of the system implies that the system is unable to differentiate between low (background) noise and the data it is attempting to process. However, when testing with SNR20, the accuracy of the system improved drastically. This suggests that the system can identify the difference between noise at 20 decibels and the desired data (the words and sentences being spoken) much better than noise at 10 decibels.

Clearly, training with only clean data to test for noisy data is not producing correct and accurate results. This shows that we need to develop a new and better system, trained using a combined set of all the clean and noisy data. The new system, employed with three Gaussian mixtures per state, will be tested separately on clean and each noisy data.

For modifications, we need to make a new list file to contain all the filenames of the clean and noisy training data and modify the global variable `$LIST_train` and `$LIST_train_HCopy` in the Perl script to call on that newly created list file. To test the data separately, we copied the same Perl script three times and modified the global variable `$LIST_test` and `$LIST_test_HCopy` to call `CLEAN1`, `N1_SNR10`, and `N1_SNR20` for test one, test two, and test three, respectively. While changing the *-p* value to find the balance between *D* and *I*, the three Perl scripts were executed, and the results are as shown in Table 3. More details on the following results are presented in Appendix 3.

Table 3: *Recognition result comparing all test data using the new system with three Gaussian model*

	%Correctness		%Accuracy	
	SENT	WORD	WORD	*-p*
CLEAN	93.23	98.73	98.27	-70
SNR10	78.13	94.57	93.64	-20
SNR20	90.1	98.15	97.69	-50

From Table 3, it is apparent that the system has been trained with more cohesive data, covering both clean and noisy data at different decibels. This has dramatically improved the results from Table 2. While the accuracy for all test results is above 90%, the correctness at the sentence level when testing SNR10 is still below the 90% threshold, implying that the system still has difficulties deciphering low noise to the sentence being said. Most likely the system, due to the low-level noise, is unable to discern when one word in a sentence has ended, and the next has begun.

Conclusion

Through the completion of this lab, we have been able to familiarise ourselves with automatic speech recognition with the use of the Hidden Markov Model Toolkit (HTK) and Perl Scripts. We have been able to look at the effect of delta-delta features on improving word and sentence recognition correctness and accuracy and how a 3 gaussian mixture model provides more correct and accurate results than a 1 gaussian mixture model. Finally, we have studied the effect noisy data can have on the recognition system, and how it is best to train the system with noisy and clean data for the most correct and accurate results when testing with clean and noisy datasets.

Appendix

1. Result comparison of employing single and three Gaussian mixtures per state

		1 Gaussian Mixture							3 Gaussian Mixtures						
		%	H	D	S	I	N	*-p*	%	H	D	S	I	N	*-p*
Correctness	SENT	88.02	169	-	23	-	192	-50	91.67	176	-	16	-	192	-40
	WORD	97.57	844	6	15	6	865		98.38	851	3	11	3	865	
Accuracy	WORD	96.88							98.03						

2. Result comparison of testing SNR10 and SNR20 using a model trained with clean data

		SNR10							SNR20						
		%	H	D	S	I	N	*-p*	%	H	D	S	I	N	*-p*
Correctness	SENT	13.02	25	-	167	-	192	-86.5	68.23	131	-	61	-	192	-57.7
	WORD	67.28	582	72	211	71	865		93.06	805	18	42	18	865	
Accuracy	WORD	59.08							90.98						

3. Result comparison of testing all data separately using a model trained with all data.

		CLEAN							SNR10						
		%	H	D	S	I	N	*-p*	%	H	D	S	I	N	*-p*
Correctness	SENT	93.23	179	-	13	-	192	-70	78.13	150	-	42	-	192	-20
	WORD	98.73	854	4	7	4	865		94.57	818	8	39	8	865	
Accuracy	WORD	98.27							93.64						

		SNR20						
		%	H	D	S	I	N	*-p*
Correctness	SENT	90.1	173	-	19	-	192	-50
	WORD	98.15	849	4	12	4	865	
Accuracy	WORD	97.69						