Automatic Speech Recognition
Project 1 Report

Team Members:

1. DUVVURU NITHISH - IMT2014016
2. PERICHERLA SEETARAMA RAJU - IMT2014038

Features Generation:

a) To generate MFCC features of train data:

1. Go to root of project directory
2. Change line 117 of import_timit.py to - timit_df.to_hdf("./features/train_mfcc/timit.hdf", "timit")
3. From the terminal, execute the command - python import_timit.py --timit= PATH_TO_TRAIN_DATA --n_delta=0 --preprocessed=False

b) To generate (MFCC + delta) features of train data:

1. Go to root of project directory
2. Change line 117 of import_timit.py to - timit_df.to_hdf("./features/train_mfcc_delta/timit.hdf", "timit")
3. From the terminal, execute the command - python import_timit.py --timit=PATH_TO_TRAIN_DATA --n_delta=1 --preprocessed=False

c) To generate (MFCC + delta + delta) features of train data:

1. Go to root of project directory
2. Change line 117 of import_timit.py to - timit_df.to_hdf("./features/train_mfcc_delta_delta/timit.hdf", "timit")
3. From the terminal, execute the command - python import_timit.py --timit= PATH_TO_TRAIN_DATA --n_delta=2 --preprocessed=False

d) To generate MFCC features of test data:

1. Go to root of project directory
2. Change line 117 of import_timit.py to - timit_df.to_hdf("./features/test_mfcc/timit.hdf", "timit")
3. From the terminal, execute the command - python import_timit.py --timit= PATH_TO_TEST_DATA --n_delta=0 --preprocessed=False

e) To generate (MFCC + delta) features of test data:

1. Go to root of project directory
2. Change line 117 of import_timit.py to - timit_df.to_hdf("./features/test_mfcc_delta/timit.hdf", "timit")
3. From the terminal, execute the command - python import_timit.py --timit= PATH_TO_TEST_DATA --n_delta=1 --preprocessed=False

f) **To generate (MFCC + delta + delta) features of test data:**
   1. Go to root of project directory
   2. Change line 117 of import_timit.py to - timit_df.to_hdf("./features/test_mfcc_delta_delta/timit.hdf", "timit")
   3. From the terminal, execute the command - python import_timit.py --timit= PATH_TO_TEST_DATA --n_delta=2 --preprocessed=False

Models Files:

./models/mfcc_2_with_energy_coeff – GMM trained on MFCC features of train data, 2 mixture components, with energy coefficient
./models/mfcc_4_with_energy_coeff - GMM trained on MFCC features of train data, 4 mixture components, with energy coefficient
./models/mfcc_8_with_energy_coeff - GMM trained on MFCC features of train data, 8 mixture components, with energy coefficient
./models/mfcc_16_with_energy_coeff - GMM trained on MFCC features of train data, 16 mixture components, with energy coefficient
./models/mfcc_32_with_energy_coeff - GMM trained on MFCC features of train data, 32 mixture components, with energy coefficient
./models/mfcc_64_with_energy_coeff - GMM trained on MFCC features of train data, 64 mixture components, with energy coefficient
./models/mfcc_64_without_energy_coeff - GMM trained on MFCC features of train data, 64 mixture components, without energy coefficient
./models/mfcc_128_with_energy_coeff - GMM trained on MFCC features of train data, 128 mixture components, with energy coefficient
./models/mfcc_256_with_energy_coeff - GMM trained on MFCC features of train data, 256 mixture components, with energy coefficient
./models/mfcc_delta_64_with_energy_coeff - GMM trained on (MFCC + delta) features of train data, 64 mixture components, with energy coefficient
./models/mfcc_delta_64_without_energy_coeff - GMM trained on (MFCC + delta) features of train data, 64 mixture components, without energy coefficient
./models/mfcc_delta_delta_64_with_energy_coeff - GMM trained on (MFCC + delta + delta) features of train data, 64 mixture components, with energy coefficient
./models/mfcc_delta_delta_64_without_energy_coeff - GMM trained on (MFCC + delta + delta) features of train data, 64 mixture components, without energy coefficient

Note:
The model for the blank phoneme is stored as BLANK.pkl

Features Files:

./features/test_mfcc – MFCC features of test data
./features/test_mfcc_delta – (MFCC + delta) features of test data
./features/test_mfcc_delta_delta – (MFCC + delta + delta) features of test data
./features/train_mfcc – MFCC features of train data
./features/train_mfcc_delta – (MFCC + delta) features of train data
./features/train_mfcc_delta_delta – (MFCC + delta + delta) features of train data

Running the code:

a) Training:

The code for the training is present In the Jupyter notebook "train.ipynb". The file has been written in a generic manner so few changes are required to generate models for different features and different number of mixture components. Only the below variables have to been changed:

timitFile = { Path to file in which train features have been saved }
numberOfMixtures = { Integer indicating number of mixture components on which the model has to be trained }
energyFlag = { 0 (With energy flag) or 1 (Without energy flag) }
directoryNameToSaveModel = {Path to directory to which model has to be saved}

Example:

timitFile = ./features/train_mfcc/timit.hdf
numberOfMixtures = 256
energyFlag = 0
directoryNameToSaveModel =./models/mfcc_256_with_energy_coeff/

Note:
For example, let "mfcc + delta" features are generated from the train data and a model is generated using 16 number of mixture components and trained on the "mfcc + delta" features with energy coefficient. During testing, to test the model, "mfcc + delta" features have to be generated from the test data and the same model has to be used with testing on the "mfcc + delta" features with energy coefficient.

b) Testing:

The code for the training is present In the Jupyter notebook "test.ipynb". The file has been written in a generic manner so few changes are required to test models generated for different features and different number of mixture components. Only the below variables have to been changed:

timitFile = { Path to file in which test features have been saved }
energyFlag = { 0 (With energy flag) or 1 (Without energy flag) }
directoryNameOfSavedModel = {Path to directory to which model has been saved}

Example:

timitFile = ./features/test_mfcc/timit.hdf
energyFlag = 0
directoryNameOfSavedModel =./models/mfcc_256_with_energy_coeff/

Calculation of Phoneme Error Rate / Question B – 4:

To get the Phoneme Error Rate, the “asr_evaluation_14.py” file from https://github.com/belambert/asr-evaluation has been downloaded and modified. The “asr_evaluation.py” file originally takes in arguments the reference text file and hypothesis text file. We modified this “asr_evaluation.py” file to take in 2 arrays as input the original labels array (reference) and the predicted labels array (hypothesis). This modified “asr_evaluation.py” file has been renamed to “asr_evaluation_modified_14.py” and placed in the root of the project directory and it is imported when used. Results of various Phoneme Error Rates are present in the Results section below. (Usage of “asr_evaluation_modified_14.py” is in the Phoneme Error Rate part of “test.ipynb” file)

Results:

a) With Energy coefficient:

Sl. No	Features	Number of Features	Number of Mixture Components	Frame Level Accuracy (%)	Phoneme Error Rate (%)
1	MFCC	13	2	12.9210	89.2211
2	MFCC	13	4	14.0304	88.0843
3	MFCC	13	8	13.8052	88.3150
4	MFCC	13	16	14.9322	87.1604
5	MFCC	13	32	14.9145	87.17855
6	MFCC	13	128	13.9815	88.1345
7	MFCC	13	256	13.6009	88.5244
8	MFCC	13	64	14.3944	87.7114
9	MFCC + Delta	26	64	18.9027	83.09226
10	MFCC + Delta + Delta	39	64	19.2549	82.7313

b) Without Energy coefficient:

Sl. No	Features	Number of Features	Number of Mixture Components	Frame Level Accuracy (%)	Phoneme Error Rate (%)
1	MFCC	12	64	13.2854	88.8477
2	MFCC + Delta	24	64	17.7824	84.2401
3	MFCC + Delta + Delta	36	64	17.8565	84.1641

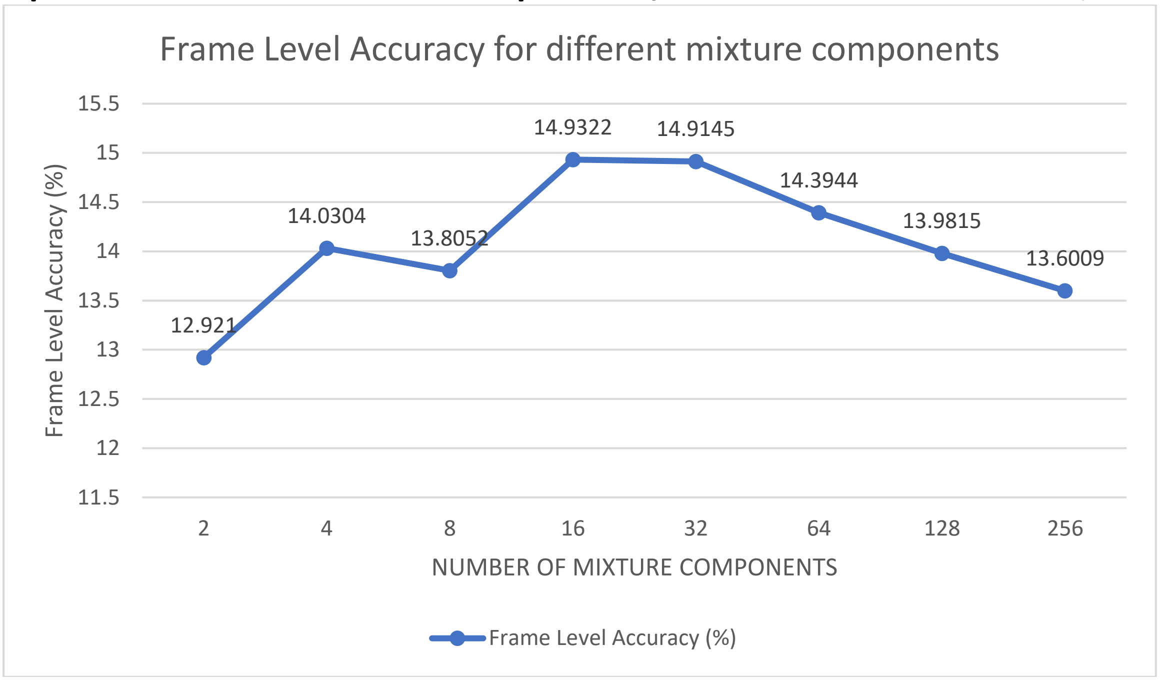
Results Discussion:

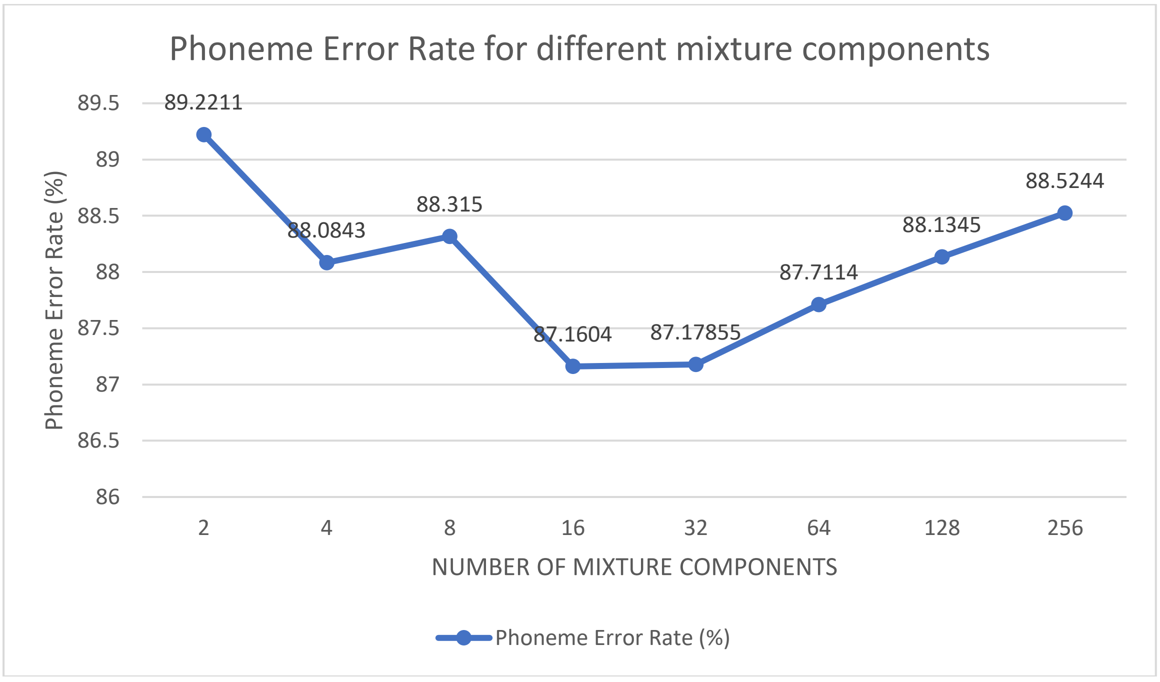
From the results table above, we have observed the following points:

- As the number of features (on which the model has been trained on) increase, the frame level accuracy increase and phoneme error rate decrease.

- The model which is trained on with energy coefficient feature produces more frame level accuracy and less phoneme error rate than the model which is trained on without energy coefficient feature.
- Observation on the number of mixture components is provided in the section below.

Optimal Number of Mixture Components / Answer for Question B – 5,6:





Graph 1 represents the corresponding frame level accuracies obtained when the test data has been tested with models trained with different number of mixture components (trained/tested with Energy coefficient of the features). From graph 1, we can observe that the frame level accuracy has showed increasing trend from “2” point to “16” point and decreasing trend from “16” point to “32” point. So, the frame level accuracy is highest when the number of mixture components has been set to 16.

Graph 2 represents the corresponding phoneme error rates obtained when the test data has been tested with models trained with different number of mixture components (trained/tested with Energy coefficient of the features). From graph 2, we can observe that the phoneme error rate has showed decreasing trend from “2” point to “16” point and increasing trend from “16” point to “32” point. So, the phoneme error rate is lowest when the number of mixture components has been set to 16.

So, the optimal number of mixture components is 16.

Note:

The TIMIT dataset has been not been uploaded in this project folder.