Accent Recognition

Input-output behaviour of system:-

The input is a .wav file and the output is a single number(0-4) representative of the accent it represents.

Evaluation Metric:-

The evaluation metric is quite straightforward in our case. There are 5 different accents that we are considering (although the dataset contains 214 accents we pick 5 from them because others have very low training data) and our classifier outputs the accent it thinks the input is from. Therefore the accuracy in this case will be
Accuracy = (# of correct predictions in test set)/(# of total predictions in test set).

Existing Approaches:-

The 2 existing approaches are from the following papers:-
a)https://pdfs.semanticscholar.org/93e9/13b002e79436042af0689bb1ac2927a3180c.pdf
b)https://arxiv.org/pdf/1602.07394.pdf

The feature extraction pipeline for both the approaches are the same :-
Speech -> Silence Removal -> Feature Extraction -> PCA -> Improved Features.
(Will tell about the details of Silence removal and Feature Extraction in the next part)

In paper a) It extracts two types of features i) Long term features ii) Short term features. The long term features were then fed to a Deep NN with the parameters as suggested in the paper(or can be seen in the answer to the next part) and the output was an 11 node softmax layer(Their dataset contained 11 accents) which would represent the probabilities of the input being in a specific class. The short term features are fed into a RNN and its probabilities for the input being in one of the classes was calculated. The two probabilities were merged by assigning weights to them corresponding to their prediction accuracies.

In paper b) The author has tried to combine phonetic knowledge in the accent recognition problem instead of just using the acoustic features. Also his feature extraction pipeline is a little bit different than the one shown above. As for the classifier he uses a GMM-UBM model which is described as follows. He used a simple fact that most identifiable accents are presented from the pronunciation of vowels rather than consonants and thus computed multiple vowel-specific GMMs with features of the vowel components

Model and algorithm

We will be following the approach given in paper a) (currently we are trying to implement the paper and later on we will factor in our own changes).
The model is as follows :-

a)First the empty spaces and the stops from the speech are removed using VAD(voice activity detection)

b) The dataset samples were cut down into 4 second segments(they were originally of size 36 seconds) hence around 8-9 segments(as some data is removed after passing through VAD) from a single sample were created. Long term features were extracted from a software called openSMILE(it's free)which gave around 6373 features for each 4 second segment. Each 4-sec window was further split into 25ms windows with a 10ms overlap. Short-term features were extracted from each 25ms signal. Specifically, we used 39 th - Order MFCC features for this task(hence there were 260*39 input features for each 4 second segment)

c)The long term features were then passed through a DNN whose structure is as follows. The input layer contains 6373 features.Three hidden layers with 256 nodes for each followed. Rectifier linear units ("ReLU") were used at the output of each layer and we use the dropout method to prevent overfitting, each input unit to the next layer can be dropped with 0.5 probability.The output layer contained 5 nodes corresponding to the 5 accents with softmax activation functions. The other hyper parameters need to be tuned.

d)The RNN was trained on the short-term features extracted from 25ms frames of speech. Categorical labels were assigned to each frame of the segment. The results for each sample were calculated by averaging the predictions on all frames in all segments. The structure of RNN is as follows: The input data is sequentially fed into the RNN frame-by-frame. Each frame is of dimension 39. Two hidden layers with 512 long short term memory (LSTM) nodes were used. The activation function for the gates was a 'logistic sigmoid' and for updating the cell state, we used a 'tanh'. The accent label was assigned to every 25ms speech frame - the LSTM layers allowed the model to learn long-term dependencies by taking the output of the previous hidden nodes as part of the inputs to the current node s. We have not trained this model yet and hence the hyperparameters have to be decided.

e)The final output will be calculated by fusing the results from the DNN and the RNN. This is done by first calculating the accuracies they get individually on the validation set(assuming an 80:10:10 split). Let them be acc_DNN and acc_RNN. Then the output probabilities for node i (i is from 0 to 4) are given by w_i,DNN*(acc_DNN/acc_DNN+acc_RNN) + w_i,RNN*(acc_RNN/acc_DNN+acc_RNN). And then the maximum among all these fused weights is given as the class(accent) of the language.

Algorithm Implemented Till Now

The dataset we wanted to use was CSLU: FAE of LDC but it requires LDC membership to access and therefore we cannot use it. Hence the dataset we are using instead is https://www.kaggle.com/rtatman/speech-accent-archive#speakers_all.csv .
It consists of 214 languages but the amount of data of various languages is very low. Hence we decided to use the top five accents which were English (559), Spanish(162), Arabic(102), Mandarin(65) and French(63).
We have played around with **openSMILE** and have managed to extract the long term features and the short term features of the data after having passed them through VAD. The variations of the following commands were used.

To remove the silence this was used :-
SMILExtract -C input.wav -C vad_segmenter.conf -waveoutput voice_segments/segment_
To extract the Long Term Features this command was used:-
SMILExtract -C config/IS13_ComParE.conf -I input.wav -O ltf.csv
(INTERSPEECH 2013 Compare challenge participants had been provided with these features)
To extract the Short Term Features this command was used:-
SMILExtract -C config/MFCC12_0_D_A.conf -I input.wav -O stf.csv

The DNN structure has also been implemented in **tensorflow** in python but the creation of batches has not been done till now. The parts remaining are creation of the RNN structure and the training of the model. There is no code available for this paper and we will be implementing everything on our own.

Experiments to be conducted

We have only found one free dataset related to our problem which was as described above(LDC2007S08 is not free and we could not find INTERSPEECH 2016 challenge dataset). After training on the five languages(and tuning the hyperparameters in DNN and RNN like lr,epochs,dropout,etc), we plan on increasing the number of languages in our training to include a few more languages. If all this works we wish to include the phonetic knowledge as given in paper b) to improve our results.