Speaker Tagged Speech Recognition

Bhavan Turaka 150050091
Bedapudi Vamsi 150050068
Rishabh Chavhan 140050061

Task

Task

- Recognizing the speaker of a given audio
- Recognizing the text spoken by a in a given audio

Methodology

Speaker Recognition

HMM Model

HMM Model

Plan:

- Model different HMM’s for different speakers
- A single HMM should correspond to a single speaker
- Keep the label of an audio file as the label of the HMM which gives maximum probability for acceptance of the audio file
- Pros: Very Fast Training
- Cons:Need a new Model for every new speaker added

Implementation details

- We model emission probabilities of each state of the HMM as a Multivariate Gaussian
- Since it has been found that the order of transitions between the different states of a HMM is irrelevant to the context of speaker recognition (while it's order is very important in Speech Recognition) we keep the transition matrix constant with equi probable transitions between all the states
- This HMM will be hypothetically equivalent to a Gaussian Mixture Model with Number of Gaussians equivalent to number of states in the HMM
- We use the EM algorithm to train the HMM with the audio files of the person it has to model

Implementation details

So the parameters that should be trained in the HMM are :

- $\boldsymbol{\Pi}$ vector of start probabilities for the states
- $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ s.t i ϵ (0,M) for the states in the HMM for emission probabilities
- So given an audio file $X = (x_1..x_T)$ the probability that it is accepted by this HMM will be

$$P(X) = \max(\boldsymbol{\pi}_i * N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, x_1)) * \Pi_{t=2}^{T} \max(N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, x_t))$$

- We compare the acceptance probabilities of all HMMs and choose the label of the one which gives best acceptance probability

Experimental Setup

- We train a new HMM for each speaker with 10 audio files of the speaker
- We train using the EM algorithm and the algorithm converges in less than 10 iterations for each speaker
- A 50 state HMM is used for each speaker
- The maximum number of speakers was taken to be 100
- MFCC vector of length 13 along with it’s delta’s and double delt’as were taken as the emission vectors per frame of the utterance with a stride of 10ms and frame length of 25ms
- LibriSpeech corpus was used for training and testing

Experiments and Discussion

Speakers vs Accuracy

Speaker	Accuracy
2	100%
4	100%
8	100%
24	100%
64	98.14%
96	93.59%

For the 96 speakers case training took 1 hour and testing took 11-12 hours

Experiments and Discussion

- While the HMM model gave very good results for speaker identification it is very difficult to use it for speaker verification
- Even when in the cases of an HMM trained on an utterance, the probability of acceptance by the HMM was very less
- The probability of acceptance depends highly on the utterance length
- So it is very difficult to set a threshold of acceptance probabbility for speaker verification
- Storing and creating a new model for every speaker we want to consider is tough
- Having knowledge of all speakers beforehand is difficult

Neural Network Model

Neural Network Model

Plan:

- Create Embeddings for Speaker Verification
- Train a Neural Network to give an embedding for an utterance
- Use an initial K-Means Clustering algorithm to determine centers and the maximum threshold of acceptance for a speaker
- Pros: Gives a direct user to vector mapping
- Cons:Need lots of data and training is slow

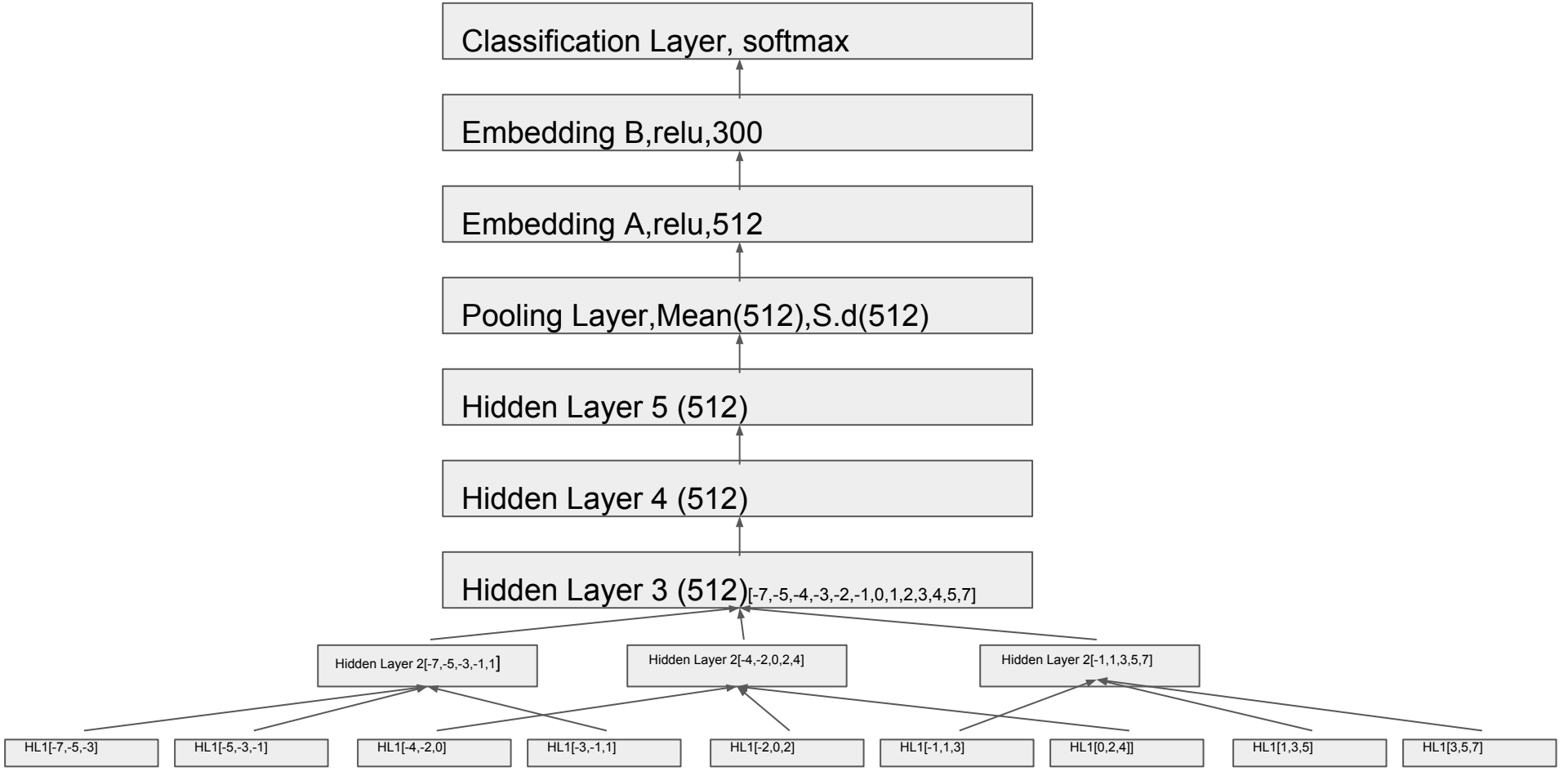
Implementation details

- We get embeddings for a user from the neural network
- Use K-Means clustering with number of centers as number of users in the training data
- Get the centers for users from the training data and a maximum allowable euclidean distance
- If an utterance is far away from the current centers make it a new user and add it to the list of current centers
- Update current centers as new vectors are added in their domain

Implementation details

Neural Network :

- We use a Time Delayed Neural Network architecture as shown
- The mfccs from time context t-7 to t+7 are used by the model
- Each utterance output is pooled by the pooling layer which is then passed to the embedding layers
- The classification layer will have the dimensions of the number of speakers in the training set
- After training is done the classification layer is removed and the outputs from Embedding A and Embedding B are used


Classification Layer, softmax
Embedding B,relu,300
Embedding A,relu,512
Pooling Layer,Mean(512),S.d(512)
Hidden Layer 5 (512)
Hidden Layer 4 (512)
Hidden Layer 3 (512)[-7,-5,-4,-3,-2,-1,0,1,2,3,4,5,7]
Hidden Layer 2[-7,-5,-3,-1,1]
Hidden Layer 2[-4,-2,0,2,4]
Hidden Layer 2[-1,1,3,5,7]
HL1[-7,-5,-3]
HL1[-5,-3,-1]
HL1[-4,-2,0]
HL1[-3,-1,1]
HL1[-2,0,2]
HL1[-1,1,3]
HL1[0,2,4]]
HL1[1,3,5]
HL1[3,5,7]

Experimental Setup

- We used a set of 100 speakers averaging up to 40 utterances per speaker
- For the output layer ‘softmax’ activation is used while relu activation is used for all the remaining layers
- The size of the layers was kept at 512 with the exception of Embedding B
- MFCC vector of length 13 along with it’s delta’s and double delta’s were taken as the emission vectors per frame of the utterance with a stride of 10ms and frame length of 25ms
- Librispeech corpus was used for training and testing

Experiments and Discussion

Speakers Vs Accuracy

Number of Speakers	Accuracy
2	50.12%
4	27.81%
10	11.2%

The accuracy only decreased with increasing number of speakers

Experiments and Discussion

- We couldn’t understand the reason for such high error rate
- It might have been for many reasons
  - The training data per person for the neural network might have been insufficient. But even when we reduced the number of neurons in all the layers (so as to account for less data) the error rates did not decrease in any way. Even we increased the number of users to increase the data supplied the error rates fell very badly
  - There might have been a bug in our implementation (we have checked and rechecked the code and tried to find out the bug in the code, if any)
  - The architecture might not have worked out .

Speech Recognition

HMM-GMM Model

HMM-GMM Model

Plan:

- Model is an Hidden Markov model constructed by composing individual Acoustic, Pronounciation, Language models
- Model is a search graph, D, using H ○ C ○ L ○ G that maps acoustic states to word sequences
- • D = πε(min(det(H̃ ○ det(C̃ ○ det(L̃○ G)))))
- Pros: Fast training for small amounts of train data compared to DNN models
- Cons: WER isn’t reducing after some level

Implementation details

- We used Kaldi LibriSpeech example and built our model over it.
- First the data, lexicon and language model data are prepared in the required format using python scripts
- Then the language model is constructed by applying kneser-ney smoothing, good turing smoothing and the one with the lowest Perplexity is selected
- Next mfccs are extracted from the train and test data
- Next the mfcc’s of train data are used for monophone training and triphone training with the help of lexicon data
- Next the accousic model created, lexicon model, language model are composed to form a search graph using optimization algorithms

Experimental Setup

- We train using the Baum-Welch algorithm and the algorithm converges in 10 epochs for monophone training and in 25 epochs for triphone training
- LibriSpeech corpus was used for training and testing

Experiments and Discussion

- WER is calculated based on number of insertions,deletions and substitutions with different insertion penalties of 0.0, 0.5, 1.0
- The WER varied for different test datas and different size of test data
- When decoded on unseen data the average WER is of 60% which we got for monophone training
- When deecoded on seen dataset the best WER is of 14% which we got for triphone training