Modeling function word errors in DNN-HMM based LVCSR systems

*Melvin Jose Johnson Premkumar, †Ankur Bapna and *Sree Avinash Parchuri
*Department of Computer Science
† Department of Electrical Engineering
Stanford University
{melvinj,ankurbpn,aparchur}@stanford.edu

Abstract—**Deep Neural Network (DNN) based acoustic models produces significant gains in Large Vocabulary Speech Recognition (LVCSR) systems. In this project, we build a DNN acoustic model and analyze the errors produced by the system, specifically the ones due to function words. We analyze the class variances of the frame data depending on whether they belong to the set of function words or not. We experiment with different ways of modeling the errors produced due to function words. Two different ways of modeling the function words in the neural network were tried and the results have been reported. We have obtained gains in the frame accuracy of one of the systems compared to the baseline. In future, we plan to build a complete system and look for improvements in the word error rate (WER).**

I. Introduction

Automatic Speech Recognition (ASR) aims to convert a segment of spoken language audio (known as utterances) to a corresponding accurate transcription. Figure 1 provides a brief overview of the architecture of a typical speech recognition system. It can be seen to consist of four main components: Feature Extraction, Acoustic Model, Decoder, Language Model.

A. Feature Extraction

During feature extraction, windows of raw PCM audio samples are transformed into features that better represent the speech content of the signal within that window. The most popularly used feature representation is Mel-frequency cepstral coefficients (MFCCs). These coefficients are derived from the spectrum of the frequency domain, representing the change in frequency content of the speech signal. The frequency domain is mapped nonlinearly so as to approximate human sensitivity to differences in pitch. Other coefficients like LPC, PNP etc. can also be used based on the application.

B. Acoustic Model

An acoustic model serves to map the extracted features to a sequence of likely spoken sounds, namely phonemes. This is usually done by using a phone likelihood estimator which can be a Gaussian Mixture Model (GMM) or an Artificial Neural Network (ANN) to estimate the likelihood of each phone. This is then coupled with the pronunciation lexicon which maps words to phone sequences. A hidden markov model (HMM) is used to model the durational and spectral variability of speech signals.

C. Language Model

In an ASR system, the language model provides the probability of a particular sequence of words. In other words, it tries to capture the properties of the language. It is used in the decoding phase along with the acoustic model to generate the word sequences for the audio signals.

D. Decoder

The acoustic model provides a distribution over all possible phonemes for each individual frame and the language model provides the prior probability of the word sequence being sensical. Using these two probabilities the decoder generates the most likely word sequence for the given audio input. As the decoder passes over the data, the possible sequences of states are stored within a graph known as a lattice, which can be pruned to reject the least likely sequences. Once the most

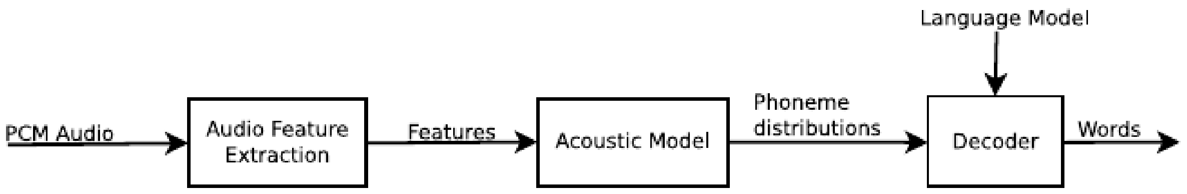

Fig. 1. ASR system architecture

likely sequence is determined, the phonemes can be mapped to complete words, creating a transcription of the original audio.

The rest of the report is organized as follows. In section II, we provide a brief overview of our experimental setup. Section III discusses our baseline system and various other experiments performed on it. In section IV, we present our experimental results. Section V provides discussion and analysis on our experimental results. We finish with the conclusion and future work in section VI.

II. Experimental Setup

The experiments were conducted using a variant of the open-source KALDI toolkit [1]. Neural-network training was done using python scripts which took advantage of GPU processing capabilities through the use of the GNUMPY and CUDAMAT libraries.

The data-set used for these experiments was the Switchboard-1 Release 2 corpus (LDC97S62), which is a collection of about 2,400 two-sided telephone conversations among 543 speakers. The data-set was divided into 300 files, each representing about an hour's worth of conversation.

The neural-net is trained using the frame-data (in the form of Mel-Frequency Cepstrum Coefficients) and the senone labels. Currently, we input 41 frames of MFCC data to obtain senone labels (from among 8986 possible senone labels) for the current frame (the other 40 frames provide context). We use the alignment files (ali*.txt), key files (key*.txt) and the feature files (feat*.bin) generated by KALDI as input to the DNN. For evaluation, we use scripts that perform a feed-forward on the neural network and output the most probable senone for the given input frame.

III. Baseline System

A. Performance of Current System

TABLE I
Error Analysis - Baseline

#wrds	%frames	%errors	frameAcc(%) (top50)	frameAcc(%) (others)
50	30.28	33.23	43.14	50.24
100	38.38	42.65	43.25	52.08
200	46.99	52.32	42.84	53.76
500	57.41	64.67	41.21	57.27

Table I shows the percent of frames containing the top *n* words, along with the contribution of the top *n* words to the frame classification errors, as well as the frame accuracy for the top *n* words and the rest of the words. As it can be seen, the percentage of errors made on the top *n* words is larger than the percentage of frames that they are present in. Additionally, the frame classification accuracy for the top *n* words is significantly lower than that for the rest of the words.

B. Experiments with epochs and number of files

Table II shows the results of the experiments run on the baseline varying the number of training files, and the number of epochs for training. While the frame classification accuracy did go up with the number of epochs, the training time also increased significantly for each additional epoch. Furthermore, the gain in accuracy was smaller as the number of epochs were raised beyond 10. Therefore, we found it more practical to use more training files (30) and less epochs (4) for our experiments, since this configuration seemed to give us a good balance

Training Files	Epochs	FrameAcc (%)
10	4	40.97
20	4	46.02
30	4	48.67
10	10	46.18
10	15	47.05

TABLE II
SENONE ERROR-RATES FOR VARYING TRAINING-SET SIZES AND EPOCHS

between frame classification accuracy and training time.

C. Top 50 words in the phoneme space

We used t-Distributed Stochastic Neighbour Embedding (t-SNE) [5] to visualize the top-50 words from the corpus in the senone-space. t-SNE is a technique for dimensionality reduction that is well suited for the visualization of high-dimensional datasets. More specifically, we used a third-party implementation of Barnes-Hut-SNE [6] which scales better to big data-sets.

Figure 2 shows the top-50 words in the corpus, plotted in the senone-space after using t-SNE to reduce the data to two-dimensions. It is seen that the top 50 function words are closely clustered together in the senone space. Hence, there are possibilities of confusions occurring between the senones corresponding to these words.

D. Most Commonly Confused Words

Table III enumerates the most commonly confused words in the baseline system. This list was generated by analyzing the original transcripts against the proposed transcripts output by the ASR system when run on a development set. The results indicate that short and frequently occurring function words like ’a’, ’the’ and ’it’ generate the most common errors. Furthermore, these function words are commonly confused for other function words that are close to them in the senone-space - for example ’a’ and ’the’ appear close together in senone space, and are one of the most frequently generated words.

Rank	Actual Word	Classified Word
1	i-	i
2	it	that
3	uh	um
4	and	in
5	in	and
6	the	that
7	uh	a
8	a	the
9	to	gonna
10	to	too

TABLE III
WORDS MOST COMMONLY CONFUSED IN THE BASELINE SYSTEM

E. Confusion Matrix for Senones

Figure 3 shows the confusion matrix for the first 200 senones in the baseline system. It can be seen that there are a lot of confusions between the senones and hence they require special attention.

F. Motivation to model function words

As observed by our results on the current system, function words form a bulk of the speech corpora. All the words in our list of top 50 words by frequency are function words. Content words are much less frequent. Our system uses MFCC labels for the current frame and neighboring 20 frames on each side to predict the senone labels for the current frame. We would expect these data vectors to have a much higher variance for content words when compared to function words, since function words have a much smaller subset of words which get repeated often while content words have a large number of infrequent words.

Our expectations are corroborated by our experiment results. We computed the class covariance matrices for the data vectors corresponding to the senones for the top 50 most frequent words and the set of content words and evaluated the eigenvalues of these matrices (to determine the gains in directions of maximum spreads). We found that the 2 largest eigenvalues for the content words are more than double those of the top 50 words [Table IV]. This indicates that the data for a particular senone is more spread out if that senone belongs to one of

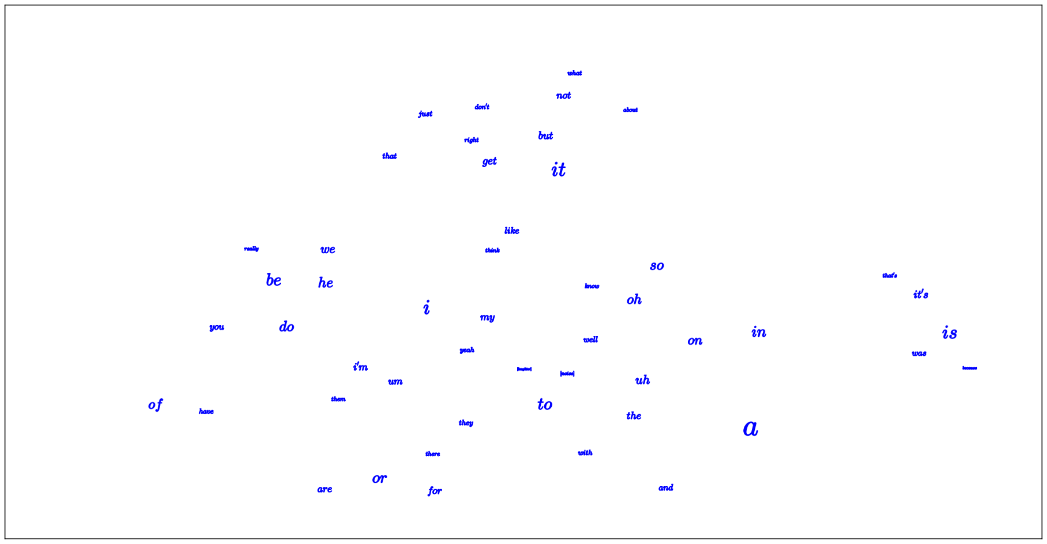

Fig. 2. Top-50 words in the senone space

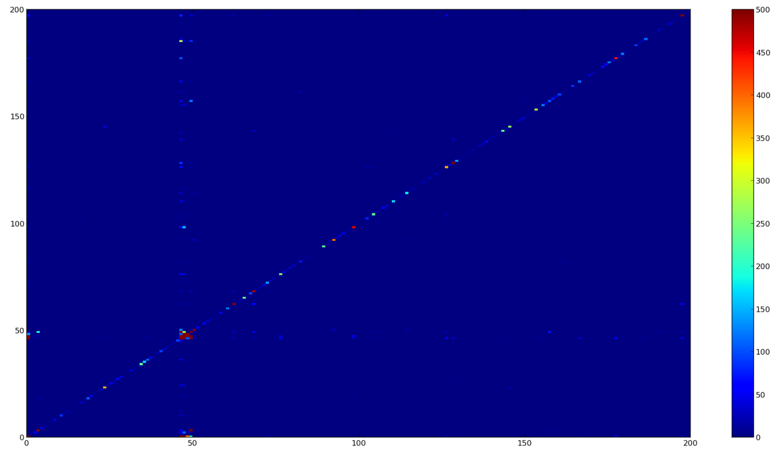

Fig. 3. Confusion Matrix for first 200 senones in the baseline system (The colormap has been restricted to the [0-1000] range to help legibility)

the content words rather than the function words.

TABLE IV
Covariance calculations

Function words		Content words	
Value1	Value2	Value1	Value2
1.68e+003	4.36e-015	3.70e+003	1.79e-013

Besides, from our experiments on the current system [Table I] we also find that the function words contribute to more errors when compared to content words. This is also observed in [4]. It was observed that during speech, function words are much less emphasized (around 14%) when compared to content words which are stressed almost 93% of the time. It was also observed that words which are stressed during speech are much less likely to get mis-recognized.

Previous work by Goldwater et al. [2] revealed that there was a significant increase in the number of errors dealing with short and frequently used words (usually function words) in Gaussian Mixture Model - Hidden Markov Model (GMM-HNN) based ASR systems. Our analysis on the DNN-HMM based ASR system revealed similar trends.

Since the senone data for function words is much less spread out and function words still contribute to a larger share of errors, we try to model their senones separately in an effort to reduce the errors produced by the speech recognition system while working with function words. We try two different approaches to model the function words senones.

In our first method, we try to create separate classes for all senones corresponding to a particular function word. This enables the system to learn from context when dealing with a function word senone. Secondly, we create a separate cluster of classes, one for each senone. This cluster is used to classify the senones corresponding to function words.

IV. Experimental Results

A. Details of the Neural Networks

The current system consists of a neural network that can be trained on the MFCC labels obtained from the speech data frames. The input to the neural network consists of the MFCC data from the current frame and from 20 frames on each side of the current frame to provide context. It outputs the probabilities of the current frame corresponding to each of the possible 8986 classes. Currently, each class corresponds to a senone (triphone). The neural network can consist of a variable number of hidden layers, depending on the experiment being performed. The frame predictions of the neural network can be fed to a HMM that has been trained on the word dictionaries. This HMM produces the sequences of phonemes which are then weighted by the language model.

After experimenting with many configurations, we finally decided to use a neural network with 6 hidden layers with 256 units in each layer. The input layer consists of 41 frames of MFCC labels and the output layer consists of varying number of neurons depending on the type of system used. A standard system takes close to 7 hours to train on a GPU and more than 30 hours to train on a CPU.

In the first system (*FunctionWordClasses*) we implement, we try to model the function words as separate classes. We define 50 new classes in the neural network, each corresponding to all possible senones for a particular function word. Our system can successfully read data from the available binary files and train the new neural network on the training set and produce the senone or class labels as its output.

In the second system (*SplitTwoClasses*) we implement, we try to model the function word senones differently from the content word senones. We create separate classes for all the senones that occur in any of the function words. This creates low variance classes for the new function word senones. Our system can successfully read data from the available binary files and train the new neural network on the training set and produce the senone or class labels as its output.

B. Implementation Details

The neural-network code was adapted from the Stanford variant of the KALDI toolkit. We wrote new data-loader instances which parsed and modified the training data as required by our models, by substituting the original senone-ids assigned to frames with the ids generated by our models.

A test harness was written to test the trained neural network using the development set, recording statistics such as frame classification accuracy, contribution of top *n* words to the frame classification error, and the frame classification accuracy for the top *n* words and the other words separately.

Additional python scripts were written to generate confusion matrices from the trained neural-network and to plot these matrices using matplotlib. Similarly, scripts were written to generate the phone-to-word and senone-to-word mappings and to visualize them using t-SNE for analysis.

TABLE V
Experimental Results

System	Frame Accuracy (%)
Baseline	48.67
SplitTwoClasses	35.07
FunctionWordClasses	52.28

V. Discussion and Error Analysis

TABLE VI
Error Analysis - FuncWordClasses

#wrds	%frames	%errors	frameAcc(%) (top50)	frameAcc(%) (others)
50	30.28	26.49	58.17	50.04
100	38.38	37.05	53.86	51.16
200	46.99	47.70	51.48	52.83
500	57.41	61.44	48.84	56.72

Table V gives the comparison between the baseline system and the two models built by us. It can be seen that the *FunctionWordClasses* model wherein a new class is created for each function word outperforms the other two models in terms of frame accuracy. The poor performance of the *SplitTwoClasses* model can be attributed to the fact that creating two versions of the same senone creates sparsity issues since we are essentially doubling the number of possible outputs. Another possible reason would be that this new scheme does not provide the context information between the senones of belonging to a single word as provided by the *FucntionWordClasses* model. We refer to the *FucntionWordClasses* model as *new-sys* and perform further analysis on it.

Table VI gives our analysis on the *new-sys* similar to the one in Table I . It can be seen that the percentage of contribution to the errors by the top n words has significantly reduced compared to the baseline system. A significant increase in the frame accuracy of the top n words can also be seen. This shows that the *new-sys* model with separate classes for each function word has been successful in modeling the errors generated due to these words.

Figures 3 and 4 represent the confusion matrices of the top 200 senones of the baseline and the *new-sys*. Figure 5 gives the confusion matrix for the newly added classes for the function words in the *new-sys*. It can be seen that the *new-sys* has much less confusions than the baseline. This bolsters the results seen in Table VI. It should also be noted that the new classes created in the new system are pretty well distributed and have comparatively lesser confusions as seen in Figure 5.

VI. Conclusion and Future Work

In this project, we aimed to model the errors caused by function words in a DNN-HMM based ASR system. We experimented with two different ways of modeling the errors. Our best system provides a significant improvement in the frame accuracy compared to the baseline system. However, these numbers are not comparable across the systems since they use different architectures. Hence, we perform a thorough analysis on both the baseline and the new system in terms of the percentage contribution of errors of the top 50 function words. We noticed a significant drop in the percentage of errors due to the top 50 function words in the new system.

In the future, we would like to integrate our best neural net with the complete ASR pipeline, i.e, editing the pronunciation dictionary, altering the HMM model and retraining it to obtain WER. We would then like to compare the WER of our system and the baseline. We would also like to try other methods of modeling the senone errors like data-driven clustering and check for improvements in performance.

VII. Acknowledgement

We are very thankful for all of the guidance, support and resources that Andrew Maas and Awni

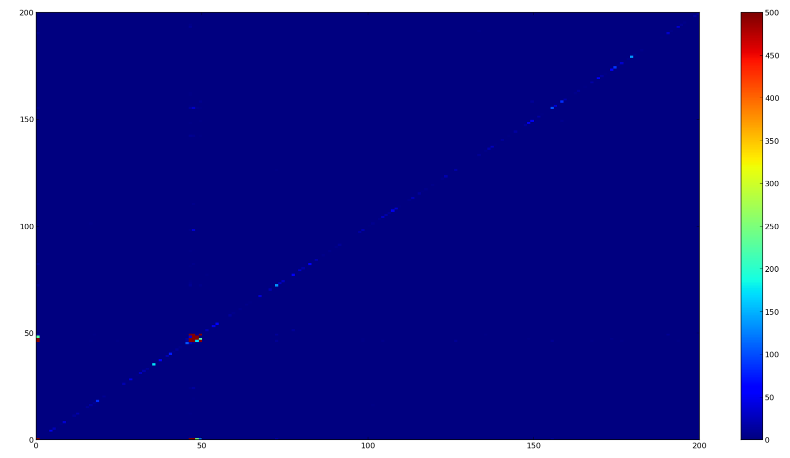

Fig. 4. Confusion Matrix for first 200 senones in the new system (The colormap has been restricted to the [0-1000] range to help legibility)

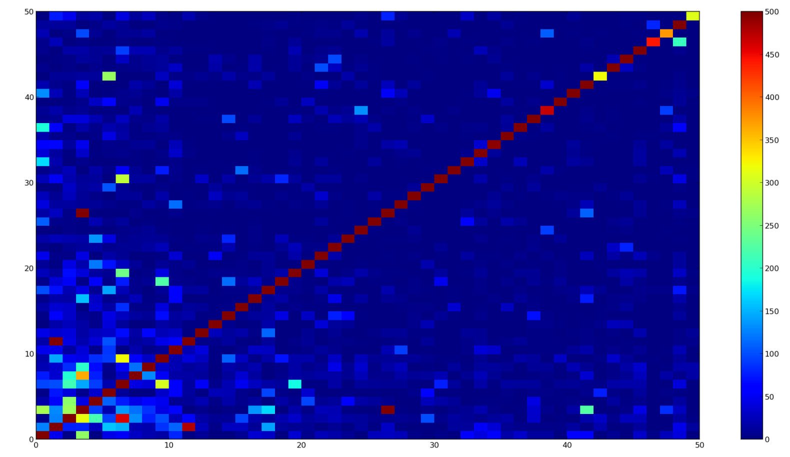

Fig. 5. Confusion Matrix for new 50 classes (The colormap has been restricted to the [0-500] range to help legibility)

Hannun have provided us through this process. We would also like to thank all the TAs for their suggestions. Finally, Prof. Manning for his reassuring words about the scope of the project and his guidance.

References

[1] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. *The Kaldi Speech Recognition Toolkit*. In IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society. December 2011.

[2] Sharon Goldwater, Daniel Jurafsky and Christopher D. Manning. *Which words are hard to recognize? Prosodic, lexical and disfluency factors that increase speech recognition error rates*. Speech Communication 52(3):181-200. 2010.

[3] Andreas Stolcke. *SRILM - An Extensible Language Modeling Tooklit*. In Proc. Intl. Conf. Spoken Language Processing, Denver, Colorado. September 2002.

[4] Alex Waibel, Kai-Fu Lee. *Readings in Speech Recognition*. Morgan Kaufmann Publishers, 1990.

[5] L.J.P. van der Maaten and G.E. Hinton. *Visualizing High-Dimensional Data Using t-SNE*. Journal of Machine Learning Research 9(Nov):2579-2605. 2008.

[6] L.J.P. van der Maaten. *Barnes-Hut-SNE*. In Proceedings of the International Conference on Learning Representations. 2013.