Proceedings of the ASME 2017 International Mechanical Engineering Congress and Exposition
IMECE2017
November 3-9, 2017, Tampa, Florida, USA

IMECE2017-71027

SPEECH ASSISTANCE FOR PERSONS WITH SPEECH IMPEDIMENTS USING ARTIFICIAL NEURAL NETWORKS

Ramy Mounir
University of South Florida
Department of Mechanical Engineering
Tampa, FL, USA

Redwan Alqasemi
University of South Florida
Department of Mechanical Engineering
Tampa, FL, USA

Rajiv Dubey
University of South Florida
Department of Mechanical Engineering
Tampa, FL, USA

ABSTRACT

This work focuses on the research related to enabling individuals with speech impairment to use speech-to-text software to recognize and dictate their speech. Automatic Speech Recognition (ASR) tends to be a challenging problem for researchers because of the wide range of speech variability. Some of the variabilities include different accents, pronunciations, speeds, volumes, etc. It is very difficult to train an end-to-end speech recognition model on data with speech impediment due to the lack of large enough datasets, and the difficulty of generalizing a speech disorder pattern on all users with speech impediments. This work highlights the different techniques used in deep learning to achieve ASR and how it can be modified to recognize and dictate speech from individuals with speech impediments.

1. INTRODUCTION

Recent advancements in Artificial Neural Networks (ANN) have allowed researchers to expect the state-of-the-art performance on every application assigned to the network. For example, prediction of depth dimension from a single camera [1] and the classification of raw pixels into objects [2].

Likewise, the advancements in Recurrent Neural Networks -especially ASR– have provided the state-of-the-art performance. The work of Graves et al. [3] in classical Temporal Classification (CTC) made it possible to use unsegmented data in an end-to-end speech recognition application [4].

Other statistical Models, such as Hidden Markov Models (HMM), Gaussian Mixture Models (GMM) /HMM, Dynamic Time Wrapping (DTW) and Deep Neural Networks (DNN)/HMM, offer a solution to the speech recognition problem [5,15]. In this work, we are implementing a Deep Bidirectional Recurrent Neural Network with a CTC output layer for end-to-end training. The chosen method provides the state-of-the-art performance in speech recognition [4] and is easier to implement with the TensorFlow framework used for designing and training this ASR model. We have not considered any of the other methods for the work presented in this paper.

Deep Neural Networks (DNN) require a large dataset for supervised training to avoid underfitting. It is challenging to find a large enough dataset containing speech impediment data to train an end-to-end speech recognition model, so we have to rely on post-processing to solve this problem.

It is important to understand the difference between a normal speech recognition system and one that is designed for persons with speech impediments. In a normal ASR system, the input sentence provided to the network is expected to have semantic meaning, and has no missing or extra phonemes/letters. This makes it easier to train a classification network and transform input speech to phonemes/letters. However, a system designed to transcribe speech impediment audio input will have to generate a number of potential words depending on the severity of the impediment, and rely on a language model to generate full sentences with semantic meanings.

This work focuses on building a full phoneme-based ASR system, modifying the output phonemes and connecting the generated potential words to a language model.

2. FEATURE EXTRACTION

The first step in any automatic speech recognition system is feature extraction. Many techniques are being used for feature extraction such as Linear Predictive codes (LPC), Perceptual Linear Prediction (PLP), MFCC, PLP-Relative Spectra etc. [6].

MFCC (state-of-the-art) seems to work best with the intended application (ASR). MFCCs were introduced by Davis and Mermelstein in the 1980's.

MFCC Feature extraction is a preparation method that helps with filtering the input audio to match what humans can hear. It extracts coefficients, for each frame, to be used as inputs to the neural network. The Mel scale (1) is used to adjust the input frequencies.

$$M(f) = 1125 \ln (1+ f/700) \quad (1)$$

Extracting the features/coefficients from an audio file involves calculating periodogram, applying the Mel filterbank and discrete cosine transform [7]. This usually results in 26 coefficients, but we dropped the last 13 coefficients because they represent fast changes in the filterbank energies and they degrade the performance of ASR. So, the result of this process is 13 coefficients for each frame of the audio.

Mel-Frequency Cepstrum Coefficients technique is clearly explained in more details in [7].

3. ARPABET PHONETIC TRANSCRIPTION

Each data point consists of an audio file (goes through MFCC process) and a text file which consists of a piece of text (matching the speech in the audio). Each word in this sentence is converted to its Arpabet phonetic transcriptions (list of phonemes describing how the word is pronounced). Then the phonemes are converted to numbers which are then called labels. So, the input to the network is 13 coefficients for each frame and the labels are list of numbers corresponding to phonemes of each word in the sentence.

The phonetic dictionary is provided by Carnegie Mellon University [8] and it contains over 134,000 words and their pronunciations using 39 (not including lexical stress markers) different phonemes. These were chosen because of their diversity and availability with massive amount of datasets.

4. ASR NETWORK ARCHITECTURE

To achieve good speech recognition performance, we have decided to implement a Deep Bidirectional Recurrent Neural Network (DBRNN) [9]. Recurrent Neural Networks (RNN) are used to share variables (Weights and biases) across time for varying length inputs/outputs. More practically, they are used when there is a need to remember inputs from the past and base decisions on them. In theory, RNNs have the ability to capture long-term dependencies, but unfortunately, they do not [10]. Vanilla RNNs also cause problems while training (Back-propagation through time) such as vanishing and exploding gradients.

Long Short-Term Memory (LSTM) cell is a modified RNN cell [11] that allows for the network to easily access long-range data from the past. LSTMs use a memory cell that can be accessed and modified using gates, which allows the cell state to remember only important details from the past and forget other information. Fig. 1 shows a single LSTM cell with peephole connections [12] (dashed lines).

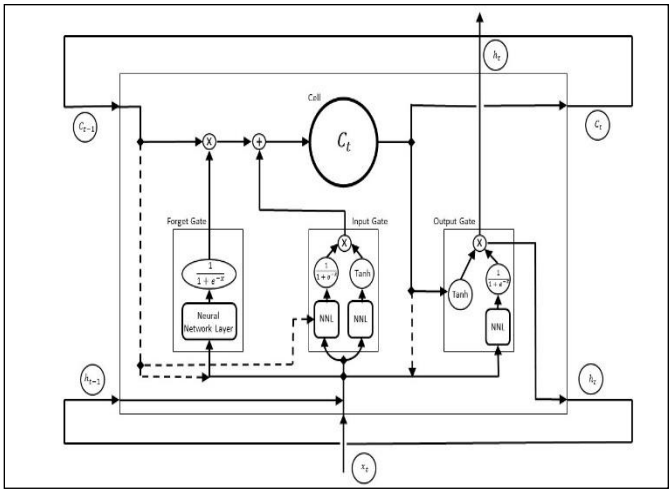

Figure 1: Single LSTM cell with peephole connections

The recurrent connection is what allows the network to share variables across time. The number of times the recurrent connection is used depends on the number of terms in the sequence being trained. In our case, it depends on the number of frames in each audio file. Fig. 2 shows an equivalent unfolded version of the LSTM cell in Fig. 1.

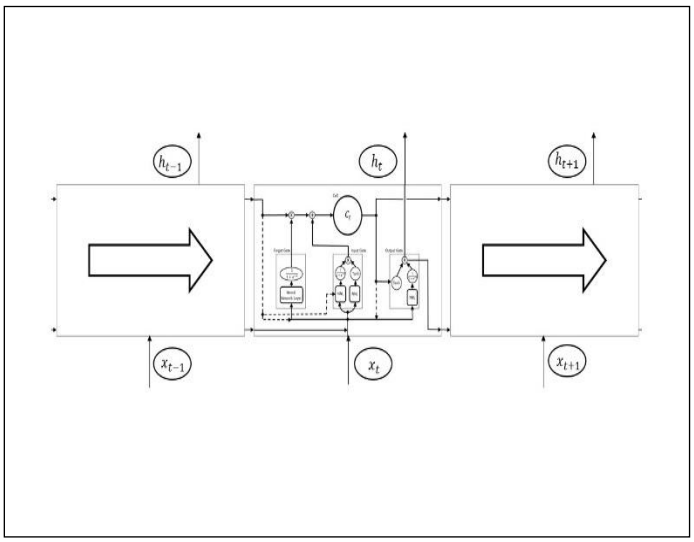

Figure 2: Unfolded LSTM Cell

Vanilla LSTM and RNNs can only access data from the past; however, in speech recognition, it is better to have access to past and future cell states at every time step. Bidirectional RNNs were first introduced in 1997 [13] and were combined with LSTMs in 2005 [14]. Fig. 3 shows a folded layer of BRNN.

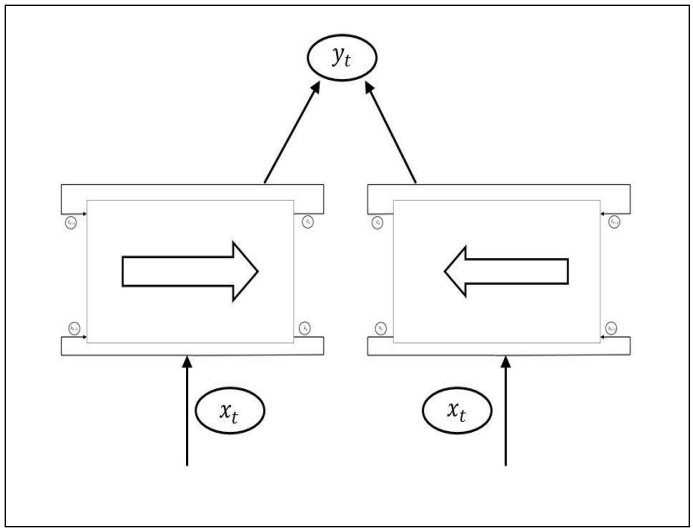

Figure 3: Folded Bidirectional LSTM Neural Network

The last modification to this architecture is to add more layers of LSTM in both directions. A deep network is better at recognizing higher level representations. It is always advised to add more layers if the network will be trained on a big enough dataset. The deeper the network, the bigger the dataset needed to train it because extra layers add more variables instead of sharing them. A Deep Bidirectional LSTM is shown in Fig. 4.

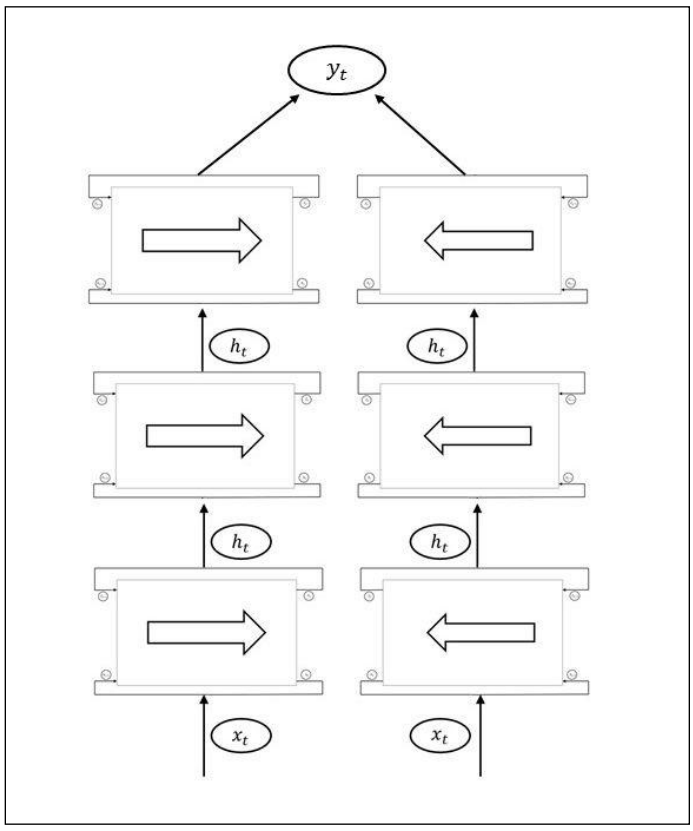

Figure 4: Deep Bidirectional LSTM

5. CONNECTIONIST TEMPORAL CLASSIFICATION

A Recurrent Neural Network requires the inputs and targets being used for training to be aligned, which makes it very difficult to find enough data for training. The input sequences (audio frames) are usually longer than the target sequence (phonemes/letters), which makes it difficult to align inputs and output. Pre-segmented acoustic data is not an option when searching for datasets.

One of the solutions to this problem is to use a hybrid DNN-HMM model. Hidden Markov Models (HMM) can work with RNNs to predict the labels [15]; however, many disadvantages result from this hybrid system that affect performance and usefulness of the model. Discussion of the disadvantages of this hybrid system is found in Graves dissertation, section 7.7 [16].

CTC [3,16] is a layer to be added at the end of a recurrent network that allows the network to predict labels at any point in the sequence, it does not require the data to be pre-segmented/aligned.

The key idea is to generate a probability distribution at every time step (frame) instead of generating a label. The next logical step is to use one of two main decoding methods; Best Path Decoding and Prefix Search Decoding [3]. Decoding the probability distribution into a maximum likelihood label is the outcome of CTC.

CTC has proved to be more accurate than the Hybrid RNN-HMM and baseline HMM. CTC is used in this work and prefix search decoding is implemented for better results.

6. ERROR MEASUREMENT

Two of the most important error measures used in speech recognition are Label Error Rate (LER) [3] and Word Error Rate (WER) [17]. The best-known way to measure the accuracy of any ASR or handwriting recognition system is to compare the labels predicted by the system to the target. LER is defined as the mean normalized edit distance between the network's prediction and the targets.

WER is similar to LER in principle. It is used to determine the error of whole words in a sentence instead of the error of labels or characters in a word. WER is usually higher than LER for ASR applications. Both error measurements methods (LER/WER) can range from zero to infinity.

7. LEVENSHTEIN EDIT DISTANCE

Levenshtein edit distance [18] is the function that calculates the minimum edit distance from one list of phonemes to another. In other words, it calculates the number of inserts, deletes and substitutions required to output the minimum combined edit distance value. Fig. 5 shows an example of calculating the edit distance of "IMECE" to "ASME" using dynamic programming.

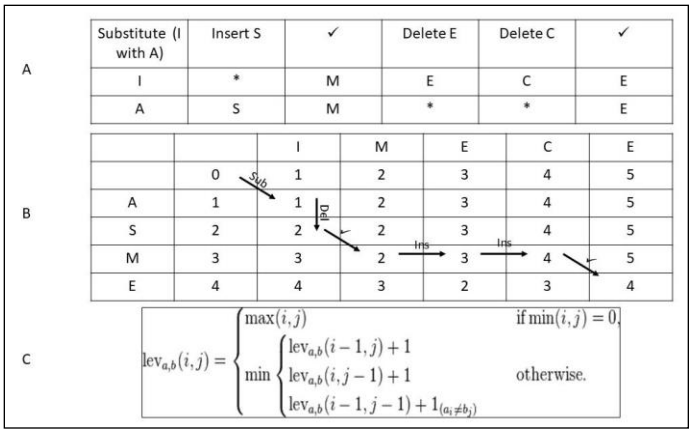

Figure 5: A) Edit operations B) Dynamic programming of Edit distance C) Algorithm from Wikipedia

8. LANGUAGE MODEL

The language model is another recurrent neural network model trained on full sentences. The model outputs the probability of a word occurring after a given word or sentence. It is simpler than the main speech recognition model because it is not bidirectional and not as deep.

The model is trained on large text files containing sentences with semantic meanings. To reduce computational complexity of the training, we use counter functions to keep most repeating x number of words and replace the rest with <UNK> token. End/beginning of sentences are also replaced with <EOS> token, this token will be used in the decoding methods to ensure that the sentences are beginning and ending correctly.

The model consists of number of basic LSTM layers (depending on the size of training data) stacked on top of each other to form a deep network. Input data are P number of words from the training batch while the targets are the same P number of words shifted by one word to the right. So, each word is trained to predict the probabilities of every word in the vocabulary (X words) to occur after a given sentence or word.

To decode the probabilities and generate meaningful sentences, we have two methods; greedy and beam search decoding [19]. Greedy is choosing the next most-probable word until the full sentence is generated. The problem with this method is that it will choose only one path based on the results at one early step of decoding. It is possible that after several steps the probability of this chosen path will decrease and a different path's probability increases. Fig. 6 shows an example of the greedy search method. Other paths like "The bottle is falling" may have had better overall probability but it was not chosen because this method only looks one step ahead and chooses a single path.

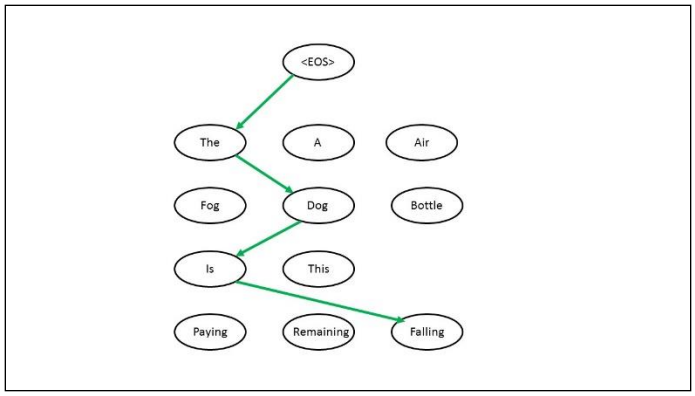

Figure 6: Greedy decoding

To solve this problem, the beam search method is used. Depending on the beam width K, the method chooses the highest K paths at each step and continues in each path individually. The result is several sentences (depending on the K value and number of steps). The sentences can be compared using the <EOS> token, by checking the probability of the <EOS> token to occur after each sentence. The sentence with the highest probability is the outcome of this method. Fig. 7 shows an example of beam search decoding. The total number of sentences is equal to K raised to the power N, where N is the number of words in the sentence (N = 4 in fig. 7).

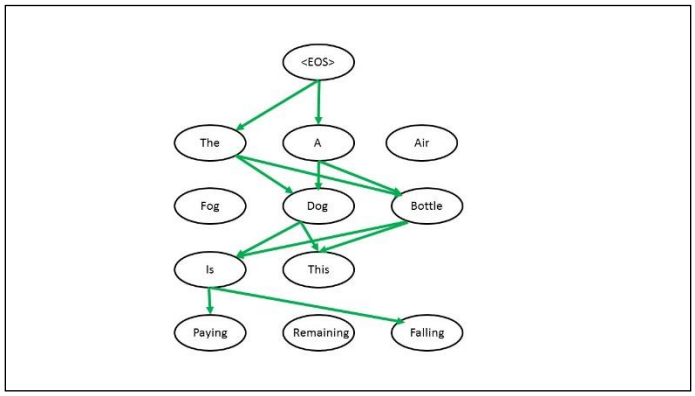

Figure 7: Beam Search Decoding

9. EXPERIMENTAL SETUP

The Bidirectional Recurrent LSTM consists of 2 LSTMs (one in each direction) with 100 hidden blocks in each direction. Then the network is made deep by adding 2 more layers of the same architecture to make 3 total number of layers. Two fully connected layers were attached to the output of the recurrent network with 128 hidden units in each.

The input to the network is (13 coefficients) x (number of frames). The input side is made dynamic; it can accept varying length of data in the same batch. The network outputs to number of phonemes (79) + blank label. Prefix/Beam search decoding was used for evaluation.

The optimizer used for backpropagation through time (BPTT) is "Adagrad" and its initial learning rate is 0.01 with a momentum of 0.95. Dropout was not used. This network was not trained on the well-known TIMIT speech corpus dataset, instead, we trained it on LibriSpeech ASR corpus [20]. This dataset consists of 100 hours of speech sampled at 16 KHz derived from read audiobooks.

Each audio file was used as 1 batch, so the number of frames in each file is equal to the number of times the variables will be shared in the recurrent network. This architecture resulted in a **38.5% LER** on the Test set.

The predicted set of phonemes (sentence) was then segmented into smaller sets of phonemes (words) using the predicted blank labels. Each word was compared to CMU phonetic dictionary phonemes to find the closest words at 1, 2 and 3 Levenshtein edit distances. Each edit distance option generates a list of words which is used by a language model to determine the final sentence (with the most semantic meaning). 5 sentences are generated (using a beam width of 2), and the sentence with the smallest edit distance is chosen.

The language model consists of a LSTM cell training in the forward direction only. The network is made deep by adding another LSTM cell to form a total of a 2-layer LSTM network. Each LSTM has 1500 hidden units and 0 forget bias. An initial learning rate of 1.0 was used.

This language model was trained on a **small** version (4,983 KB) of the Penn Tree Bank (PTB) dataset containing 10,000 distinct words. The model was trained for 55 epochs. For decoding, a beam search method was used with the options of 1 (greedy) and beam width of 2.

We prepared another language model for testing, with a training dataset 30 times larger than the PTB dataset used above. We downloaded and pre-processed 500 random e-books [22] to form a 150,000 KB training dataset. This model contains 30,000 distinct words and was trained for 2 epochs.

The LER and WER is calculated for all the options mentioned above (edit distances and beam widths). A total of 100 random test data points (limit of 6 words in each sentences) were used from the testing dataset and the averages of results were calculated for each option. Results from using both language models for LER and WER are shown in section 10.

Additionally, we have tested the full system with 8 different human subjects, generating 2 data points from each test. The subjects were asked to record a sentence and use a scale from 1 to 10 to answer survey questions, such as how close the output phonemes are to the actual phonemes they said and how good the system identified spaces between the words. We were able to calculate how many words in the spoken sentence were found in the potential words' lists at each edit distance. Results are shown in section 10.

10. RESULTS AND DISCUSSION

Table 1 shows the LER and WER, including standard deviations for ASR + PTB dataset. Table 2 shows the LER and WER, including standard deviations for ASR + downloaded eBooks model. As shown in results, the increase in beam width has resulted in a decrease in the error (as expected). The increase in edit distance resulted in an increase in the error because the test audio files do not include audio with speech impediment.

Table 1: LER + ST. DEV and WER + ST. DEV. for ASR + Small PTB language model

Beam Width	Edit Distance 1	Edit Distance 2	Edit Distance 3
1	55.3 ± 13.6% / 100.3 ± 25.9%	70.4 ± 12.6% / 108.7 ± 22.6%	76.2 ± 10.1% / 110.0 ± 20.5%
2	52.0 ± 13.5% / 97.6 ± 28.8%	64.4 ± 11.4% / 105.4 ± 24.3%	72.0 ± 8.9% / 108.2 ± 21.4%

Table 2: LER + ST. DEV and WER + ST. DEV. for ASR + downloaded eBooks language model

Beam Width	Edit Distance 1	Edit Distance 2	Edit Distance 3
1	59.5 ± 18.0% / 96.9 ± 28.0%	67.3 ± 14.1% / 101.2 ±27.5%	75.4 ± 13.8% / 103.6 ± 26.1%
2	53.6 ±17.1% / 88.5 ± 31.1%	63.1 ± 13.0% / 99.4 ± 27.6%	71.1 ± 12.4% / 102.9 ± 25.9%

The model generates more words when the edit distance is increased. If the audio does not include any speech impediment, searching at an edit distance of two will result in more error than at one. *Our target is to use the smallest edit distance that guarantees the delivery of the correct words to the language model.*

Figure 8 shows how many words per sentence were found by the test subjects in the potential words at each edit distance. Test subjects having no accents were able to find significantly more words, at an edit distance of one, than subjects with accents. As we increase the edit distance, the words/sentence found increase for all the data points. This concludes that it is recommended to increase the edit distance for data with speech impediment to acquire better results (given a good language model). The number shown after each accent in the legend represents the data points count. Figure 9 shows the survey results on the phoneme accuracy, space detection and the application's ease-of-use.

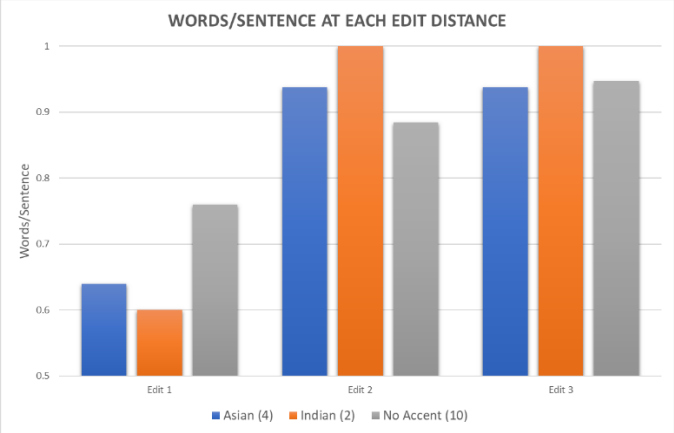

Figure 8: Words found per sentence at each edit distance for all test subjects

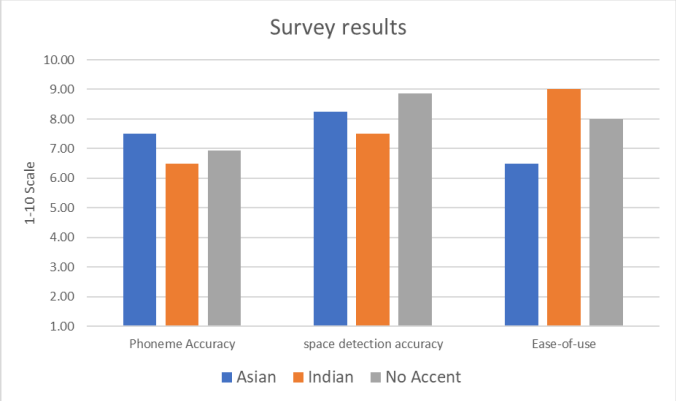

Figure 9: Test subjects' opinion on various aspects of the ASR system

The edit distance and beam search width are options that can be modified to achieve different performances to assist persons with speech impediments. Increasing the edit distance and beam width can be translated into relying more on the language model than on speech recognition.

The idea of relying more on the language model and less on speech recognition seems to be comparable to how the human mind tries to understand distorted audio. We know that the speech must have semantic meaning so we generate words close to what we hear by changing the phonemes and connecting them in a way that makes sense grammatically and logically. The amount of words we generate from each distorted word depends on the severity of distortion that we expect from the audio.

In other words, if the audio file contains major speech impediment, the model will not find the correct words within an edit distance of 1; however, the model has a higher probability of finding those words at an edit distance of 2 or 3. It is always better to use a larger beam width, especially when a large edit distance is used.

In practice, the user can adjust the edit distance to match the severity of speech impediment, then adjust the beam search to a value that provides the best results at a reasonable computational complexity (time consuming process).

11. FUTURE WORKS

These results presented in this paper can be improved by using better datasets. The TIMIT dataset should be used here for better performance because the data points are shorter (less frames per run), which helps the model learn better.

The language model should be trained on the full PTB dataset for a significantly better performance. Also, the eBooks dataset should be trained for more than 2 epochs and the architecture can be modified into a deeper (more layers) and wider (more hidden units) network. We are planning to include testing data with speech impediment in the experiment for future work.

12. CONCLUSION

In this paper, we have implemented a complete speech recognition model using recurrent neural networks and connected it to a language model. We also presented options that can be modified to assist persons with different levels of speech impediments. The presented results matched our expectations for the effect of the edit distance and beam width on the LER/WER, and we are working on including test data with speech impediment to further improve the models and prove the validity of the discussed concepts.

ACKNOWLEDGMENTS

The authors would like to thank the Florida Department of Education - Division of Vocational Rehabilitation for their support.

REFERENCES

[1] D. Eigen, C. Puhrsch, and R. Fergus, “Depth Map Prediction from a Single Image using a Multi-Scale Deep Network,” http://papers.nips.cc. [Online]. Available: http://papers.nips.cc/paper/5539-depth-map-prediction-from-a-single-image-using-a-multi-scale-deep-network.pdf.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Advances in Neural Information Processing Systems, 2012.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification,” Proceedings of the 23rd international conference on Machine learning - ICML '06, 2006.

[4] A. Graves and N. Jaitly, "Towards End-to-End Speech Recognition with Recurrent Neural Networks,". [Online]. Available: http://jmlr.org/proceedings/papers/v32/graves14.pdf.

[5] “Speech recognition,” Wikipedia, Available: https://en.wikipedia.org/wiki/Speech_recognition#Models.2C_methods.2C_and_algorithms.

[6] N. Dave, “Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition,” INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY, vol. 1, no. VI, Jul. 2013.

[7] P. Cryptography, "Mel Frequency Cepstral Coefficient (MFCC) tutorial," in practicalcryptography.com. [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[8] C. M. U., “The CMU Pronouncing Dictionary,” The CMU Pronouncing Dictionary. [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[9] A. Graves, A.-R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013.

[10] Y. Bengio, P. Semard, and P. Frasconi, “Learning Long-Term Dependencies with Gradient Descent is Difficult,” IEEE Transactions on Neural Networks, vol. 2, no. 2, Mar. 1994.

[11] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, 1997.

[12] F. Gers and J. Schmidhuber, “Recurrent nets that time and count,” Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, 2000.

[13] M. Schuster and K. K. Paliwal, “Bidirectional Recurrent Neural Networks,” IEEE Transactions on Signal Processing, 1997.

[14] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” Neural Networks, vol. 18, no. 5-6, pp. 602–610, 2005.

[15] D. Yu and L. Deng, Automatic speech recognition: A deep learning approach. United Kingdom: Springer London, 2014.

[16] A. Graves, “Supervised Sequence Labelling,” Studies in Computational Intelligence Supervised Sequence Labelling with Recurrent Neural Networks, 2012.

[17] M. Thoma, “Word Error Rate Calculation,” Martin Thoma, 15-Nov-2013. [Online]. Available: https://martin-thoma.com/word-error-rate-calculation/.

[18] M. Gilleland and M. P. Software, “Levenshtein Distance, in Three Flavors.” [Online]. Available: http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm.

[19] D. Heres, “Sampling Strategies for Recurrent Neural Networks,” 30-Aug-2016. [Online]. Available: http://danielheres.space/posts/2016/08/30/sampling-strategies-for-recurrent-neural-networks.html.

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books,” 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015.

[21] A. Taylor, M. Marcus, and B. Santorini, “The Penn Treebank: An Overview,” Treebanks Text, Speech and Language Technology, 2003.

[22] “Free ebooks by Project Gutenberg,” Project Gutenberg. [Online]. Available: https://www.gutenberg.org/wiki/Main_Page.