CS230

Automatic Speech Recognition for Childhood Language Development

Daniel P. Ryan
Department of Computer Science
Stanford University
dryan2@stanford.edu

1 Problem Description

Language development problems have been known to occur in up to 25% of young children.[1] Although this is typically just the normal variation in childhood development, it is often a source of frustration for both the parents and the child. Also common with young children, is an attraction to handheld computing devices and digital images. In this paper, we propose an Automatic Speech Recognition (ASR) implementation that provides a fun, interactive tool for language development. As the child speaks into the device, the algorithm identifies potential word matches and displays images associated with those words. When the child completes a word, the image associated with the successful word is displayed along with the word itself. If no word is identified, the process starts over. We anticipate that potential word matches will be obtained from a word bank of simple, common words, and that this word bank can be expanded as a child's language progresses. The input to the algorithm will be spoken audio and the output will be a list of potential words and their associated confidence scores.

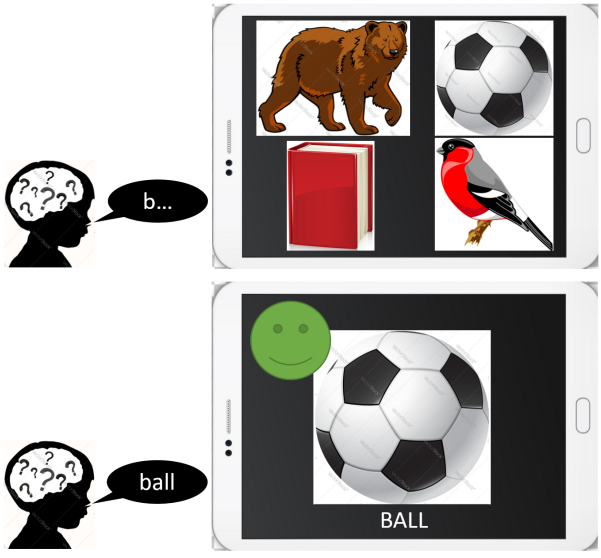

Figure 1: Pictorial of ASR tool for language development

Due to time and resource constraints, the full implementation of this tool has not been realized on this project. The word detection cannot yet be performed with "live" audio, rather pre-recorded audio

CS230: Deep Learning, Spring 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

must be passed to the model. Additionally, when keywords are recognized, images associated with those words are not displayed to a user. Rather, a time series of integers representing the keywords is output to a file. The developer can then lookup the keyword associated integers in a word bank and compare to the expected output.

Some of the challenges associated with this tool are that the model must be robust enough to be able to accommodate children's limited verbal skills and ability to properly enunciate words. Additionally, audio quality may be a challenge that will have to be overcome. If implemented, one could imagine a child holding the device in such a way that the microphone was obstructed, dramatically reducing audio quality.

2 Dataset

Our originally selected dataset, the Open Speech and Language Resources (OpenSLR) LibriSpeech SLR12 corpus [2], evaluated using the ESPnet end-to-end speech processing model, proved to be unsuccessful. After spending many long hours attempting to execute the ESPnet model, our efforts were unsuccessful. Furthermore, while the LibriSpeech dataset represents a large corpus of data suitable for a general purpose speech recognition model, the model for our application is reflective of a much more specific purpose. Rather than having the ability to identify thousands of words, our model only needs to recognize a very small sample of words, on the order of tens of words. For these reasons, we have chosen to build our dataset from the Tensorflow Speech Commands dataset [3], specifically designed for keyword spotting systems. The dataset consists of 100,000 utterances of 35 words, with each utterance stored as a 1 second, or less, WAVE formatted file. Additionally, a few larger samples of background noise are provided. From this, we selected 6 words that we would consider positive keywords (bird, cat, dog, one, two, three), 6 negative words (bed, down, left, seven, wow, yes), and 5 samples of background noise, 10 seconds in length. Similarly to how we generated a dataset in the Coursera Trigger Word Detection programming assignment, each training sample was generated by selecting one background sample and overlaying a random number of positive words, and a random number of negative words. While the samples were being assembled, we also created labelled output files. Whereas the Trigger Word Detection assignment only required a binary labelling scheme, we now use a multi-class labelling scheme to identify the specific positive word injected. The development and test datasets were manually generated by recording audio on a Logitech webcam and hand labelling to provide a more representative sample of what would be expected under real world operating conditions. Due to the amount of effort required to hand label samples, these datasets were much smaller than the training dataset. The training set consists of 2000 samples while the development and test sets consist of 26 and 5 samples, respectively.

This approach provided a straightforward method to align the labels of the dataset at the word level, as opposed to at the sentence level, as it is in the LibriSpeech dataset. This was an additional reason that the Tensorflow dataset was selected for use.

3 Architecture

Primarily, this work will focus on implementing a unidirectional recurrent neural network (RNN) model. RNNs are a powerful tool for sequential data and thus are a natural choice for analysis of temporal data. While traditional neural networks assume that the inputs are independent of eachother, RNNs utilize the entire sequence of inputs to perform predictions. The following related literature was also be utilized for inspiration and guidance: [3], [4], [5], [6], [10]

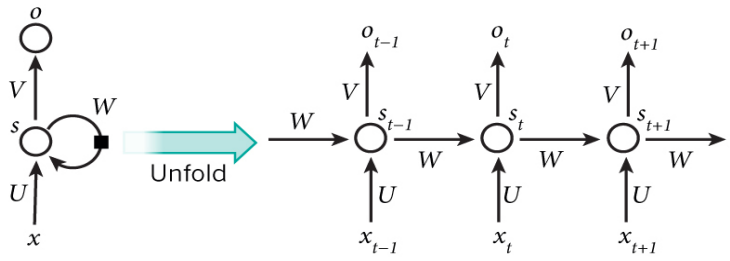

Figure 2: Pictorial of Recurrent Neural Network[7]

As mentioned in the previous section, this work was modelled after the Trigger Word Detection programming assignment and extended to accommodate mulit-class word labelling. The core of the model was implemented with the Keras API running on the TensorFlow machine learning platform (`https://keras.io/about/`). The model consists of an initial 1D Convolutional layer with a window length of 15, a stride length of 4, and consisting of 196 filters. Following the Convolutional layer are intermediate Batch Normalization, ReLu Activation and Dropout layers. The next layers in the stack are two Gated Recurrent Units, representing the core of our RNN model. Each layer is followed Dropout and Batch Normalization layers. Following that, is a Time Distributed Dense layer with sigmoid activation and the number of units equal to the number of classes in our labelling scheme. Finally, the last layer is a SoftMax Activation layer.

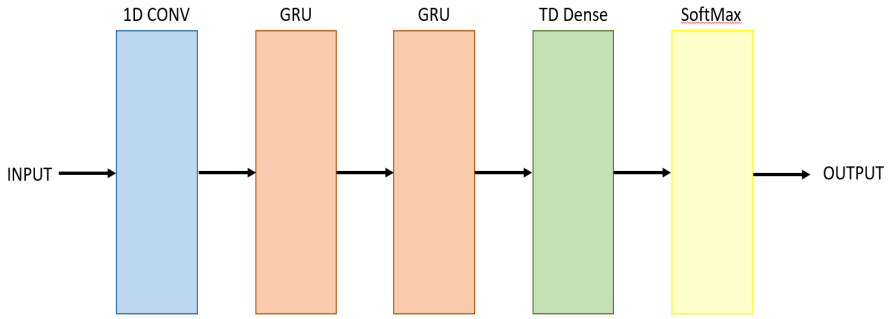

Figure 3: RNN Model

The Softmax Activation layer provides our classification by outputting a value between 0 and 1 for each of our word classification, effectively a probability that a specific word has been recognized. The probabilities, including for the no word or unrecognized word, must sum to 1 at each point in the time series.

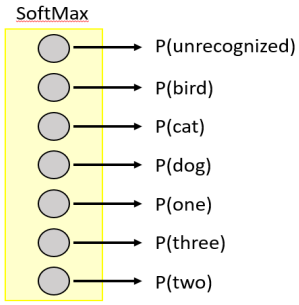

Figure 4: Softmax Activation

The primary parameters utilized to affect the training speed and accuracy are the batch size and number of epochs to train over. The training stage of the pipeline takes these parameters as input and allows us to iterate faster before running time-consuming training stages.

4 Results

Initial results show some promise, but also leave much room for improvement. The model provides reasonable accuracy on both the train and development set, with values of around 90%. However, these results are diminished because at most of the output samples in time there should be no word detected. Although the model performs correctly in this respect, a better metric is does the model predict the correct word when a word is detected. To get a better sense of how our model is performing, we examined the samples in our testing set and displayed them graphically to determine where our model was under-performing. Given that our test set was small, this manual process was not too tedious. Looking at these results, we can now start to see the true performance of the model. We can see that the model does a good job of not activating on the unknown words, and does activate on several of the positive keywords. However, it does a poor job of identifying which keyword has been detected and also fails to detect several other keywords.

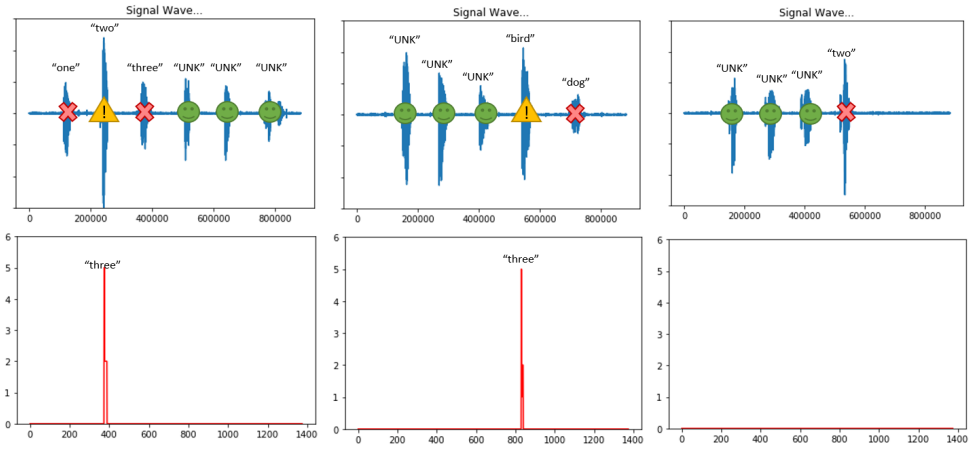

Figure 5: Test Set Results

5 Analysis of Results

Based on the results shown in the previous section, it is clear that there is much room for improvement in our model. It is possible that the model could benefit from more training. Our best model to date ran for several hours on a GPU based machine but we were still seeing improvement in our training accuracy we can assume that there is some marginal improvement to be made by longer training. A more strategic approach may be to expand the training and development datasets and add more varied examples. Both datasets are relatively small, by deep learning standards, and it is reasonable to assume that a model trained on a larger dataset might perform better. Additionally, the training dataset was very limited on the number of negative word examples provided, with only 6 used. In practice, and in our development and test sets, the model should be expected to handle any number of previously unseen words, or even word fragments or general sounds. Finally, we assume that our model itself could be made larger and more complex by adding additional layers. Given that the accuracy on our development set has not yet plateaued, we have not overfit to our training set and increasing the size or complexity of the model should be a reasonable approach.

6 Discussion

Although we did not achieve the results we had hoped for with our model, this project was enlightening in the challenges of Automatic Speech Recognition and the potential for deep learning applications.

In retrospect, our initial approach to work with a more complicated dataset (LibriSpeech) and to attempt to get a complicated off-the-shelf model (ESPNet) to work was a mistake and cost us a lot of time. Our time would have been better spent by first developing a very simple model of our own, getting that model to function, and then working to improve that model.

7 Contributions

This project was completed individually by Daniel Ryan. Any open-source material that was used has been documented in the code. The author would like to thank the CS230 teaching assistants, Jo Chuang and Jonathan Lingjie Li, for their help and guidance throughout this project.

8 Github Repository

All code created for this project can be found at the Github repository linked below. The repository contains all source code, the required data, and pre-built scripts to execute each stage of the pipeline. For execution instructions, follow the README file.

`https://github.com/dan-ryan21/cs230_ASR-for-Childhood-Language-Development`

References

[1] Trubo, R. (2020, April 20). Helping Your Late-Talking Children. Retrieved April 20, 2020, from Grow by WebMD: https://www.webmd.com/baby/features/helping-your-late-talking-children#1

[2] Panayotov, V., & Povey, D. (2020, April 20). LibriSpeech ASR Corpus. Retrieved from Open Speech and Language Resources: https://www.openslr.org/12/

[3] Yu, D., & Deng, L. (2015). Automatic Speech Recognition: A Deep Learning Approach. New York: Springer.

[4] Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., . . . Acero, A. (2013). Recent Advances in Deep Learning for Speech Research at Microsoft. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 8604-8608.

[5] Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE.

[6] Graves, A., & Jaitly, N. (2014, January). Towards end-to-end speech recognition with recurrent neural networks. In International conference on machine learning (pp. 1764-1772).

[7] Britz, D. (2015, September 17). Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs. Retrieved from WildML: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[8] Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., ... & Renduchintala, A. (2018). Espnet: End-to-end speech processing toolkit. arXiv preprint arXiv:1804.00015.

[9] Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209.

[10] Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006, June). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning (pp. 369-376).