Summary

Recurrent Neural Networks (RNNs) are a class of neural networks that can “predict” the future. This ability to “predict” is due to their structure of looking like a feedforward neural net, but that also have connections that point backwards. RNNs use the output from x-1 as an input for x to help predict x. In language modeling, RNNs can be used to model language and to analyze sentiment within reviews for movies, restaurants, service, etc.

Research Design

For this assignment, we will look at two sets of movie reviews – 500 positive and 500 negative. We will conduct a 2x2 factorial design experiment, looking at using different pretrained word vectors, as well as adjusting the vocabulary size to see how much more accuracy we can achieve in classification when using a larger vocabulary size. We use the movie reviews and vectors to classify whether or not a movie review has a positive or negative sentiment.

Technical Overview

In order to conduct the classification experiment, we will carry out a factorial design of two levels on two experimental factors, as stated earlier. We also will be making use of recurrent neural networks as they are a powerful class of neural nets, often used in sentiment analysis. We will use the Python chakin package to obtain GloVe embeddings and use two pretrained word vectors. The word vectors used for this experiment were Glove.6B.50d (50 dimensions, 400,000 corpus vocabulary size) and GloVe.Twitter.50d (50 dimensions, ~1,200,000 corpus vocabulary size). Since it can take a while to train larger vocabulary sizes, we will compare a vocabulary size of 500 vs. 400,000. Four RNN models will be compared against each other: Model A –

GloVe.6B.50d (500 words), Model B – GloVe.6B.50d (400,000 words), Model C – GloVe.Twitter.50d (500 words), Model D – GloVe.Twitter.50d (400,000 words). For classification, we will compute the softmax cross entropy between logits and the model will be optimized using the Adam Optimization Algorithm (similar to the classical stochastic gradient descent) because it is computationally efficient. For RNN hyperparameters, we used a batch size of 100 and 50 epochs. The experiment will be conducted using a Python API for TensorFlow, which is an open source machine learning library, initially developed by Google. The metrics we will be comparing will be the training and test model performances.

Results

From the train/test model performances, it is very apparent that models with larger vocabulary sizes yield higher train and test results (Model B and D). Model D yielded the highest in both train (0.85) and test (0.685), so a larger vocabulary size at 400,000 words and the pretrained vector of GloVe.Twitter.50d is recommended for higher model accuracy performance.

When looking at using a language model to classify written customer reviews and call and complaint logs, it is recommended to use RNNs to help identify positive/negative customer feelings. We would want to use preprocessing our raw data into numerical vectors to then feed into RNNs and to look into using a large vocabulary size (even seeing if accuracy converges after a certain amount of vocabulary size). At first, it may be expensive/labor intensive, but it would be beneficial if during the data gathering information stage if we could go into and identify as many logs/reviews with positive/negative feelings. We might also want to especially note terms used in negative feelings to help detect within an automated customer support system. If the automated system can help flag negative reviews/logs, it can make customer service a lot more efficient in addressing customer pain points.