D213: Performance Assessment Task 2

Part A: Research Question

1. One question an organization could ask is if it is possible to determine customer sentiment based on the review that they wrote.
2. The main goal of this analysis will be to see if we can find any relationship between word choice along with review length to see if we can determine if a customer has a good review or a bad review, thus giving us insight on the product and how the product performs and what could be improved.
3. While there are many forms of text classification neural nets, the one that is being used for this project will be a convolutional neural network, since that’s the one being used with keras and tensorflow to tokenize the words and give proper analysis and understanding to them as well as finding the relationships between them.

Part B: Data Preparation

1. For this text dataset, I ended up creating a list that prints out all the unique words. After scanning through them, you can find that there are interesting words that don’t really qualify as words, such as “gx2” or “bt50”. These may be typos or a description of the product, we just aren’t sure. We can also find words like “cent’s” which isn’t the correct way to word something, but it demonstrates part of the difficulty of analyzing text. One thing we can gather though, is that there are 5271 unique ‘words’ within this set that the tokenizer is identifying as an actual word. The stop words were added which lowered the vocabulary size to 5127 for the 3 datasets of amazon, yelp, and IMDB sentiment pages. This is still quite a few words to analyze so we took the top 1000 vocab and tokenized those for analysis. The full tokenized set is attached.
2. The purpose of the tokenization process is to break apart sentences and words into smaller parts to be analyzed. The process helps to get quantities and frequency patterns of words by isolating them based on cases, characters, and punctuation. Attached is the code to demonstrate this process.
3. The padding process is used to make the sentences being analyzed a similar length to try and cut out superfluous “junk” sequences to create an easier analysis as well as eliminating outlier characters and words. The padding will add zeroes to a sequence to make them longer to reach the required length, while deciding the max number of words required for an analysis. It will also truncate sentences that exceed the max word count. See attached for examples of padded sequences
4. This analysis will only have 2 categories of sentiment. It’s going to be positive or negative based on the positive or negative rating of the review given.
5. The first thing that was done to clean the data was to turn the data into csv files for analysis. I took all the text sets and reduced them into 2 columns of the review as well as if it was a positive or negative sentiment. I then combined all of them into a single large dataset for easier analysis and to help get a better understanding of positive and negative sentiment.
6. See attached for the combo test and train dataset along with the tokenized dataset.

Part C: Network Architecture

1. For my tensorflow analysis I used the internal library keras to produce a summary of a flattened model or a global model. Using the flatten model we can see that there are 341,197 parameters that are being analyzed for the neural network which has been flattened into a single layer that will have 640 outputs. My global model will have a few less parameters of 337,741 and will have 64 outputs. I have also taken the liberty of attaching a plot model for my flattened and global model to help visualize the neural network being created.
2. The number of parameters in my model is 341,197 for my flatten model and 337,741 for my global model. For my model there are 4 layers that are being analyzed. My sequential model has an embedding layer, followed by a flatten layer (for which makes this model unique) and 2 dense layers with a final output layer. My global model is similar, since it has an embedding layer, global layer, and 2 dense layers also. The unique difference is the flattened and global layers which offer different amounts of neuron connections to form a difference neural net for analysis.
3. For this project, we looked at various hyperparameters in this neural network to estimate actual parameters that might be needed for a better estimate of our model. An activation function that I used for my dense layers were rectified linear and sigmoid functions. The number of nodes per layer were ranging from 10 to 640 depending on if I was using a flattened model or a global average model. The loss function shows very little loss after only 1 epoch in this analysis. To avoid overfitting, it was only running for 5 epochs to generate a loss function that appears to demonstrate an accuracy of 99.64% with a loss of 0.000067 which gives a very high accuracy with our analysis. To optimize this function, we looked at our loss function graph. It appears to perhaps be an overcorrection of the function that is to direct on the text analysis and might not allow for too much wiggle room when observing or analyzing different patterns of text or word choices. Our stopping criteria for this analysis was simple time constraint. We didn’t want to overfit our curve or be too vague that we couldn’t get anything out of it. However, the neural net took over 1 hour to run only 5 epochs for a training set which was very detrimental to our understanding of the dataset. The overcorrection that is present within my analysis most likely is due to the inability to train my set within a reasonable time frame. Given an extra chunk of time or a better way of training my neural net, I could help fix the overcorrection, but it’s currently not possible within this analysis. To evaluate this model, I looked at the accuracy that the model produced during its epochs. However, this seems odd since after the first epoch it was registering an accuracy of 100%, which gives a clear indication of overcorrection. But like stated above, this is limited by physical/hardware constraints.

Part D: Model Identification and Analysis

1. A general stop criterion is better than using pure epoch counts because epochs can sometimes cause overcorrection or even under train models which could cause issues with our natural language analysis. The stopping criteria for this model would help so we find that perfect balance between over and under correction. Plus, by using epochs we have a higher likelihood of finding specific iterations in our neural network for specific cases in and how many iterations we would need. For this example, we ranged for about 1000 epochs because this number changes based on the statistical backing behind the stochastic nature of the algorithm or evaluation.
2. I grabbed the first few epochs from my model instead of trying to show all 1000 that were used. The attached code shows the various training plots and graphs
3. According to my model, my fit is 100%. This is a clear indication that there is overfitting being done. I tried to take account of this by physically changing the number of epochs that are done as well as messing with the neural nets on the backend to see if I could have my tokenized grid be mapped better, but the unfortunate issue I ran into is just having the computational power to analyze these problems in a reasonable amount of time. My over correcting was most likely a result of too large of a batch used for my analysis. I ended up scrapping a good bit of the data and recreated the tokenized dataset to analyze it. I ended up lowering the vocabulary count to the top 1000 and tried basing my epochs on the stopping criteria of halting before being overfitted which seems to have popped out a more reasonable accuracy of 94.2% which is pretty good by most metrics.
4. My predictive accuracy seems to be 100% as stated above, but this is most likely an over correction. If I test my model it seems decent at predicting the sentiment of a review, but it is unclear how it’s coming to its conclusions. After tweaking the epoch counts and the word tokens, I was able to get a 94.2% accuracy. The stopping criteria that were used was the tensorflow keras library which checks for improvements in the data and when the improvements cease to occur, it stops the sample which this stops the analysis from getting over corrected which was the original cause of our 100% accuracy.

Data Summary and Implications

E. See attached for code, neural network, and all other aspects of this project.

F. The neural network that I used/modified seems to function correctly. Since the RNN is used, it causes the network to cycle back on itself to help store learned data in memory to learn similarities in what makes a review sentimental. This uses the training and test data together to create a more complete view of the data that is being analyzed. The way this neural network behaves is the looping back function within the RNN allows it to learn from itself and creates a multiple layer deep neural network that allows for different weights to be attributed to certain words or phrases allowing the network to make decisions and correct itself and statistically make better choices on what words and meaning are trying to be conveyed in a review. This allows us to figure out the sentiment in a review with relative ease.

G. For this analysis it would be incredibly difficult to recommend a course of action without a better ability to analyze data. The original question that was proposed was, “can we determine customer sentiment from review data?” and after this analysis, I’d say it’s possible. With the realm of natural language processing, it seems that using sentimental analysis along with an RNN would allow us to classify these processes. We can determine with a 94.7% accuracy if a remark is positive or negative based on our tokenized dataset that we trained using our model. While the implementation of a feedback loop is out of the scope of this project, using the analysis presented would allow us to add a few more modifications to judge if words were positive or negative in connotation. If I had to make a recommendation to the stake holders, it would be that perhaps another method of analysis should be used in tandem with this analysis to get a better understanding of the sentiment being used. Perhaps a convolutional neural network could be used to help analyze the data as well. This would be able to provide extra insights onto our sentimental data while also creating more of a neural net with more layers to help justify our decision.

Video:

Panopto video: <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8ebb3be7-6b46-4160-a2b5-ae290183ea37>

Sources:

<https://www.kaggle.com/code/arunmohan003/sentiment-analysis-using-lstm-pytorch/notebook> , Sentiment analysis using LSTM – PyTorch, Arunmohan\_003

<https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/> , A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks, Jason Brownlee

<https://www.ibm.com/cloud/learn/recurrent-neural-networks> , Recurrent Neural Networks, IBM

<https://www.analyticsvidhya.com/blog/2021/06/nlp-sentiment-analysis/> , Starters Guide to Sentiment Analysis using Natural Language Processing, Nikhil Raj

<https://cnvrg.io/sentiment-analysis-python/> Getting Started with Sentiment Analysis using Python, Ahmad Anis

<https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/1%20-%20Simple%20Sentiment%20Analysis.ipynb> Simple Sentiment Analysis, bentrevett

<https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/> Use Early Stopping to Halt the Training of Neural Networks At the Right Time, Jason Brownlee