Sentimental Analysis on Twitter Data using RNN

Raymond B. Adams

December 2019

Abstract

Sentiment analysis has become one of the most important methods among companies. It is a way for companies to understand how their consumers feel about their product and brand as a whole. This paper discusses the methodologies used to create a sentiment analysis in Python by using a deep recurrent neural network. These methods include loading data, preprocessing data, tokenization, word embedding, and building the RNN model. This paper also discusses the results after compiling the model. The model does well but there is still room for improvement.

1 Introduction

Many companies study user data to help improve their businesses. A lot of information can be extracted from consumer data. Because this information can tell companies so much, many topics on data analysis have been created. Sentiment analysis is one of the fields that has gotten a lot of use over the years. Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative or neutral [9]. The incoming message (input) is a string or group of strings. While the sentiment (output) is usually a value, 0 for negative and 1 for positive. This machine learning task is so important to companies because it gathers public opinion behind certain topics. Knowing the sentiment allows companies to make improvements around what the public believes is lacking.

2 Related Work

This paper references four scholarly articles to help explain the procedure of sentiment analysis and the results of this project. The first article is titled, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining" is written by Alexander Pak and Patrick Paroubek. This article focuses on Twitter which is a microblogging platform. Pak and Paroubek explain how to collect a body of text, otherwise known as a corpus. They then provide their process behind building a sentiment classifier for determining whether the text is positive, negative, or neutral. Pak and Paroubek state that their methods and techniques are better than prior approaches. The second article “Deep Learning With TensorFlow: A Review” is written by Bo Pang, Erik Nijkamp, and Ying Nian Wu. This article covers the architectural decisions of TensorFlow. This paper also discusses the overview of convolutional neural networks, otherwise known as CNN. The third article titled, “This Is How Twitter Sees The World: Sentiment Analysis Part One” is written by Ronald Wahome. This article covers the methods behind preprocessing text data for sentiment analysis. The last article titled, “Turning Words Into Consumer Preferences: How Sentiment Analysis Is Framed in Research and the News Media” is written by Cornelius Puschmann and Alison Powell. This article contends that the presumptions placed into sentiment analysis as a method are followed by several constraints. This is concerning sentiment analysis’ mechanical limitations and theory. Puschmann and Powell then demonstrate a short analysis of typical framing of sentiment analysis in the new media, featuring the assumptions that exist concerning its analytical potential.

3 Dataset and Features

The data sets used in this project are comprised of tweets, movie reviews, and crawled reviews that came from different shopping sites. The first data set is the train set. This is a CSV file that contains users’ comments and sentiment values of 1,282,902 rows in one column. This column separates the text from the value by a semicolon. The value accompanied with the text is either a 0 or 1. The 0 represents a negative comment and the 1 represents a positive comment. Figure 1 shows a bar chart of the total count of zeros and ones.The second data set contains 162,426 rows of comments all contained in one column. This is also a CSV file. However, this file does not contain a sentiment value. But it does contain a semicolon at the end of every text. The semicolon is there to allow the machine to predict the value. Preprocessing was done on both the training and testing sets in the Jupyter notebook. There are a group of words that are referred to as ”stop words” in Natural Language Processing. The most common ”stop words” are ”is”, ”are”, and ”have”. These words should not be learned by the NLP because they have no connection with sentiment and will not contribute or hinder the accuracy of the model. Thus, removing them will save processing time. This code also removes symbols that are not useful such as the period, comma, black slash, forward slash, exclamation point, and semicolon. The features used were ‘tweet’ and ‘sentiment’. The target variable, ‘sentiment’, was set in the code at the end of the preprocessing section.

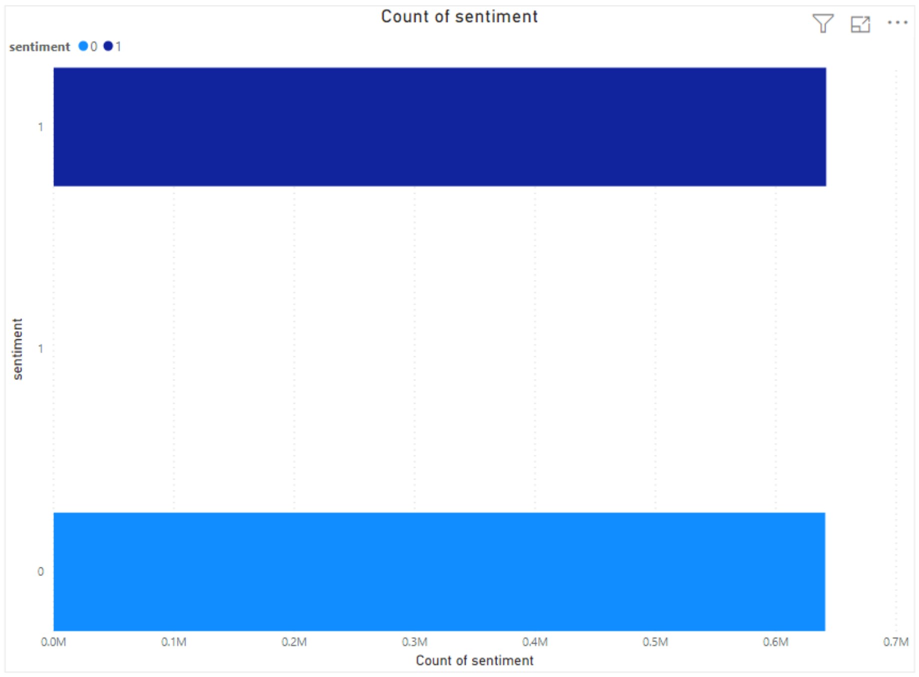

Fig. 1: Sentiment Count Displays the total amount of tweets with negative (0) or positive (1) sentiment

4 Methods

The first step inside of the jupyter notebook was to load the data. The data was loaded using pandas python library. Before the data was processed the text needed to be cleaned. The text was converted into a list. The strings were then stripped of all symbols. Symbols are not useful in determining the sentiment of a text so it is better left out. The text was also converted to all lowercase. The next step in the code was tokenization. A recurrent neural network requires the data to be in sequences. Keras Tokenizer() function converts the text in the train and test datasets to sequences. These sequences are then put through embedding matrices. Figure 2 shows the most common negative words in the dataset while figure 3 shows the most positive. The next step was word embedding. Word embedding is the natural language processing technique that maps words to a n-dimensional space. It is used to teach computers how words are used and related. In other words, it is a way to represent words to a neural network. The final step was building the RNN model.

The model learns through a deep neural network, specifically a recurrent neural network. Recurrent Neural Network is a generalization of a feedforward neural network that has internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation [3]. This network consists of an embedding layer, bidirectional CuDNN LSTM and GRUs, convolutional layers, batch normalization, and dropout layers. A deep neural network is an artificial neural network that has numerous layers between the input and output layers. A network is considered to be deep the more layers it has. The more layers a network has the more anxious it is to learn specific facets of the input data.

A favorable use of deep learning is word embedding. Embedding is a method used to represent discrete variables as continuous vectors [1]. In other words, embedding is how the neural network understands words and their relationships with other words. Neural network embeddings are useful because they can reduce the dimensionality of categorical variables and meaningfully represent categories in the transformed space [1].

Bidirectional LSTMs train two LSTMs on the input sequence. The first LSTM is put on the input sequence just as it is but the second is on a reversed duplicate of the input sequence. Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory [3]. An LSTM is trained using back-propagation. LSTMs resolve the problem of vanishing gradients in recurrent neural networks. That is why it is implemented in this project's algorithm. Next comes the convolutional layer. The convolution layer is the first layer to draw out features from an input. Convolutional layers are often used on image data but this project uses it for text data. Nonetheless, it does not matter that the text is not an image. The text (input) just needs to be thought of as an image of width n and height 1. Tensorflow provides a conv1d function that does that, but it does not expose other convolutional operations in their 1d version [6].

Lastly, batch normalization and drop out layers are implemented. Batch normalization is used to stabilize and perhaps accelerate the learning process. It does so by applying a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1 [2]. The last step is dropping layers. Dropout layers are used to combat against overfitting. After the final dropout, the model is compiled and fit.

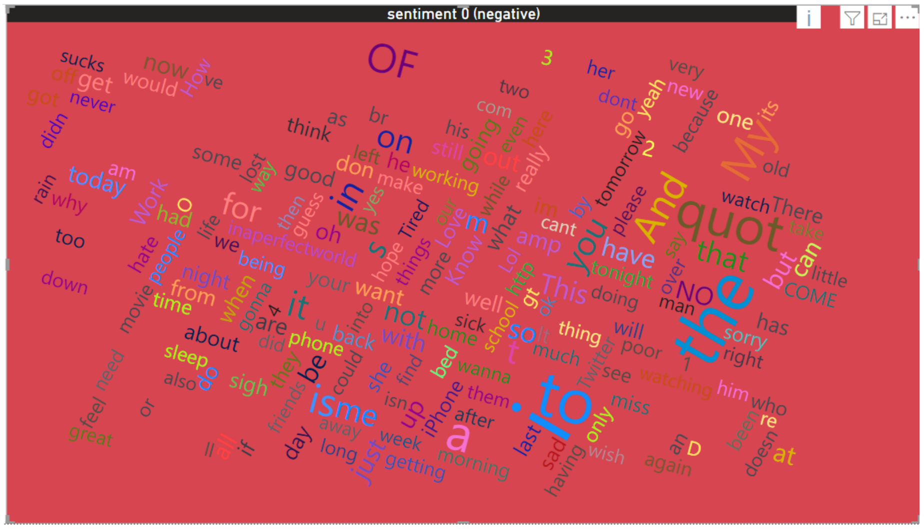

Fig. 2: Word Frequency Displays the most frequent negative words. The larger the word the more frequent it is.

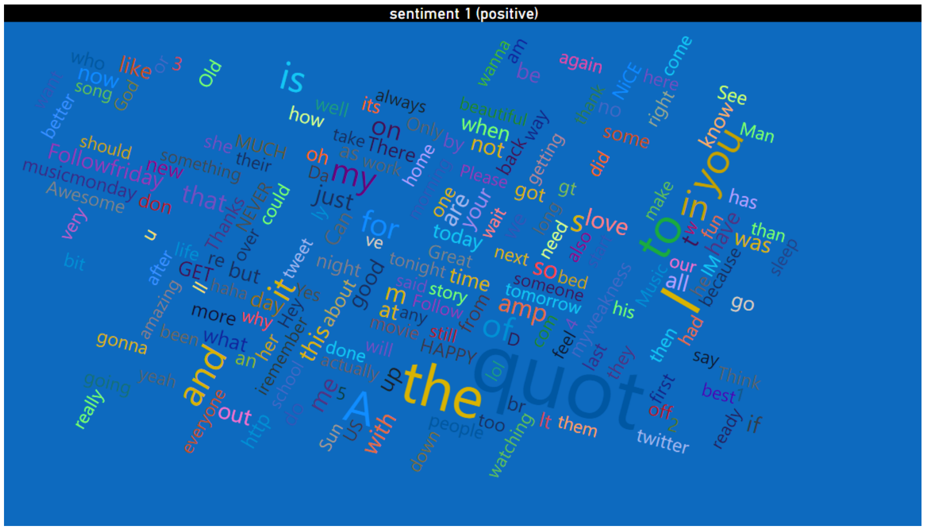

Fig. 3: Displays the most frequent positive words. The larger the word the more frequent it is.

5 Results

The model was run with a learning rate of 0.01 and a decay rate of 0.001. The learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect to the loss gradient. The lower the value, the slower we travel along the downward slope [8]. The decay rate is used to prevent the weights from increasing too much. This model was run with 150 units. The unit is a parameter in the model that refers to the dimensionality of the hidden state and output state. The next parameter is spatial_dr which is an abbreviation for spatial dropout. It was set to 0.5 for a 50% dropout rate. This is used to prevent overfitting. This works by dropping the inputs to a layer. The kernel size was set to 4. This refers to the width by the height of the filter matrix. Thus the matrix is 4x4. The parameter dense_units refers to the output shape. In this project, the dense units were set to 64. The parameter dr hold the input value for another dropout method in the model. This dropout was set to 0.2. The last parameter convolves the output to a dimension of 32 by 32.

When this model was run it performed 10 epochs. An epoch refers to one revolution through the entire training dataset. In layman terms, if you think of the model as a student, one epoch would represent one lesson that the student attends out of however many total classes there are. The model was trained on 1,026,321 samples and validated on 256,581 samples. This means that 1,026,321 strings of text and sentiment values of 0 or 1 were run through the model. Then the model tested itself on 256,581 strings of text and tried to predict the correct sentiment value. Because there were so many training and validation samples one epoch took on average 11.08 minutes to run. In total it took 110.8 minutes to complete all ten epochs. This time possibly could have been cut down had the code been run on google collab.

The first epoch also had a validation accuracy of 0.7566. While the tenth epoch had a validation accuracy of 0.7711. This value could have been increased had the model run more epochs and changed the rate at which the model learns. However, the epochs were not increased for this project because 10 epochs took a long time to run. Figure 4 displays the model accuracy that was plotted after the model was done running.

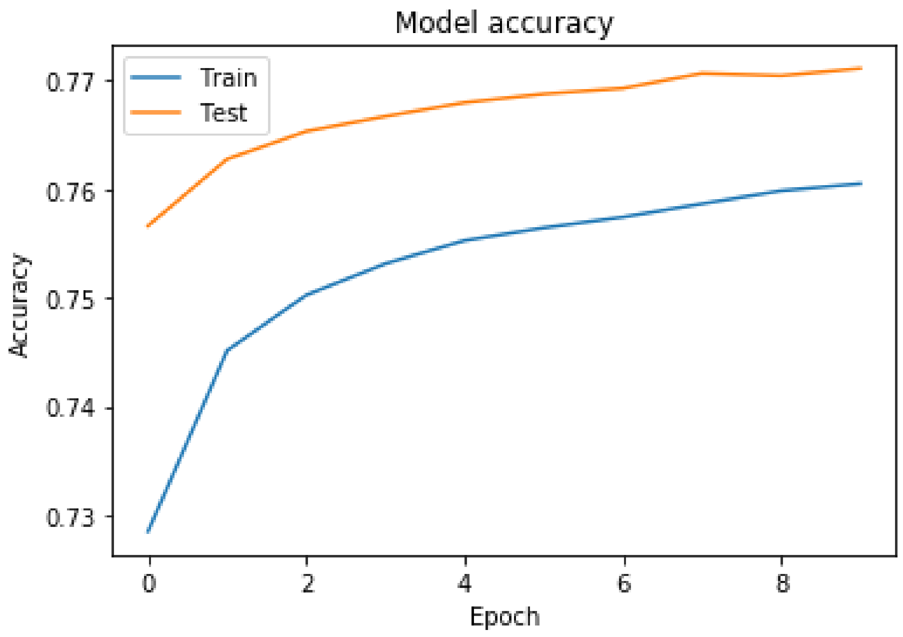

Fig. 4: Displays the accuracy of the training and validation set

6 Conclusion

To conclude sentiment analysis is a very useful method. It allows companies to examine their user's feelings and beliefs towards their brand and merchandise. The data set for this project contained two CSV files. One file was for training the data and the other was for testing the data. Both data sets contain symbols that needed to be extracted from the text. These texts were then converted into sequences. Word embedding was then used to convert the sequences into vectors of numbers. Lastly, the model was run. The results from running this model show that the model was well trained and able to correctly validate 77% of the data as a negative or positive sentiment. If this code were to be improved google collab could speed up the computational process, thus allowing more epochs to run. Running more epochs potentially can increase the accuracy of this model.

References

[1] Koehrsen, W. (2018). Neural Network Embeddings Explained. [online] Medium. Available at: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526.

[2] Maklin, C. (2019). Batch Normalization Tensorflow Keras Example. [online] Medium. Available at: https://towardsdatascience.com/backpropagation-

and-batch-normalization-in-feedforward-neural-networks-explained-901fd6e5393e.

[3] Mittal, A. (2019). Understanding RNN and LSTM. [online] Medium. Available at: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e.

[4] Pak, Alexander Paroubek, Patrick. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. Proceedings of LREC. 10.

[5] Pang, B., Nijkamp, E. and Wu, Y. (2019). Deep Learning With TensorFlow: A Review. Journal of Educational and Behavioral Statistics, p.107699861987276.

[6] Perry, T. (2017). Convolutional Methods for Text. [online] Medium. Available at: https://medium.com/@TalPerry/convolutional-methods-for-text-d5260fd5675f.

[7] Weerkamp, W., Carter, S. and Tsagkias, M. (2011). How People use Twitter in Different Languages.

[8] Zulkifli, H. (2018). Understanding Learning Rates and How It Improves Performance in Deep Learning. [online] Medium. Available at: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

[9] Gupta, S. (2018). Sentiment Analysis: Concept, Analysis and Applications. [online] Medium. Available at: https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17 [Accessed 30 Nov. 2019].

[10] McKinney, W., others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[11] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science amp; Engineering, 9(3), 90–95.

[12] Oliphant, T. E. (2006). A guide to NumPy (Vol. 1). Trelgol Publishing USA.

[13] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[14] Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

[15] Abadi, Mart39;in, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., . . . others. (2016). Tensorflow: A system for large-scale machine learning. In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16) (pp. 265–283).

[16] Chollet, F. (2015). Keras.