Sentiment Analysis of Reviews using Recurrent Neural Networks

Debapratim Chakraborty

August 2020

1 Introduction

Sentiment analysis, which harnesses the principles of aspect-based opinion mining, is quite challenging in the field of computing due to the subjective nature of words in sentences and their contextual variances. I have overcome the contextual variances of words by using Recurrent Neural Networks to predict the sentiments from the reviews. I have applied various correction measures to clean the data to enhance the accuracy of our predictions.

The newly classified predictions can be used for a variety of purposes including :

1. Coming up with the overall rating of a store/product.
2. Building the profile of a user that can be used to build up a recommender system later.

The motivation behind doing sentiment analysis on text reviews is to be able to enhance the recommendation systems that we use by taking into account not just the numeric rating but also the textual reviews given by users related to a product/business or service.

2 Premise

"What other people think" has always been an important piece of information for most of us during the decision-making process. Long before awareness of the World Wide Web became widespread, many of us asked our friends to recommend an auto mechanic or to explain who they were planning to vote for in local elections. But the Internet and the Web have now made it possible to find out about the opinions and experiences of those in the vast pool of people that are NOT our personal acquaintances.

However, with an average over 200 reviews on a particular product, no user has the time to actually go through each text review while trying to decide whether or not a product/business would be to his liking. As a result of which people tend to go with the overall numeric rating which is computed as an average of all ratings given by users. In general consumers do not use all available information and are more responsive to quality changes that are more visible(such as overall numeric rating). Consumers also respond more strongly when a rating contains more information.

Websites like Yelp do filter out users based on quality,number of friends they have,the number of reviews that they make; however while computing the overall rating of a business they take the mean of the numeric rating of only the "elite" yelp members who are filtered out by Yelp's own recommendation system. This leaves a large amount of information unused, namely the textual product reviews. This information has a significant impact on the way a customer thinks about a product and whether or not they actually use that product. I aim to extract aspect from the data and mine the opinions and enhance the recommendation process that one has been always relying on.

3 My Approach

3.1 Data Gathering

From Kaggle Website, the Yelp data sets have been used. First, the different review texts have been organised on the basis of individual unique tags based on category. So our output files have a lot of data and review texts based on different categories like sports, medicine, education, restaurants etc.And under each such file, based on business ID, we can identify which place or business is being referred to in the reviews. I have only used a part of the data - i.e - the reviews of various restaurants for this job at hand.

3.2 Data Cleaning

Steps Followed in Data Cleaning:

1. Assign a Positive (1) tag to a review if it has rating greater than 3 else a Negative(0) tag - called 'Sentiment'.
2. Check if the 'Sentiment' classes are balanced.
3. Convert all 'Review' to lowercase, strip off html tags, punctuation, and stopwords.

3.3 Data Preprocessing

Steps Followed in Data Preprocessing:

1. Tokenize each 'Review', and encode the words to a sequence.
2. Find the average length of the review (say 'k').

3. Pad all the sequences (i.e reviews) to length 'k'. Those with greater no. of words will be truncated. Those with lesser number of words will be padded with a reserved token - 0.

4. Randomly split the data. Keep 0.25 part as test data [not to be used in training]. From the training set, take out 0.05 part for validation [to be used during training].

3.4 Building the Model

I have used sequential model from Keras, and have filled it with Embedding, LSTM, Dense and Dropout layers. Using the correct dimensions for word embedding is perhaps the most important part. The embedding size is kept at 32 (gave best results) and the vocabulary and input length are calculated from the data.

The model was trained for 4 epochs. More epochs gave rise to overfitting (correlation between training and validation metrics got disturbed beyond 4 epochs).

Predictions were made on the previously split test data.

4 Checking the Results : Predicted Data

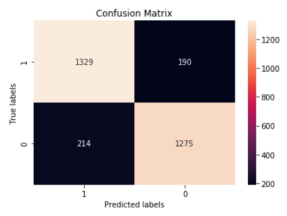

Figure 1: Confusion Matrix

Metrics :

1. Accuracy : 86.57%
2. Recall : 87.49%
3. Precision : 86.13%
4. F1 Score = 86.8

5 Conclusion and Scope

It is worth highlighting that the tensor representation of dataflow in training the model significantly reduces computation time. Performing Word Embeddings are a tricky operation made simpler with the use of RNN's.

The test results obtained are good enough to let us deploy this model in real life scenarios. We can build upon this model to enhance the overall rating or score of a particular business or product based on all the textual reviews it has obtained. This can also be effectively used to build the profile of a user and get to know what he or she likes or dislikes. We can then build a recommendation system to suggest outlets or services he or she might be interested in.

The source code with a detailed explanation and analysis can be found here.