­

**Advanced Data Analytics**

**Task II: Sentiment Analysis**

Stephen E. Porter

College of Information Technology, Western Governors University

Dr. William Sewell

February 12, 2023

**Advanced Data Analytics, Task II: Sentiment Analysis**

In this paper, I will use the provided data sets containing customer reviews for Amazon products, IMDB (movies/television), and Yelp. Using Python and it’s related packages, I will clean, prepare, and analyze the data set using a Recurrent Neural Network (RNN) to analyze word trends in positive and negative reviews. The goal is to create a model that is able to accurately predict the rating (Positive or Negative) based on the words used in the review and their order.

# Part I: Research Question

## A1. Research Question

Given a dataset of customer reviews and ratings of various products from Amazon, TV/Movies from IMDB, and businesses on Yelp, can a model be created that can learn the patterns of words in the review to predict the corresponding rating value? By so doing, we can gain better insight into customer experience across multiple areas (products, entertainment, businesses). Through extension, similar companies that offer review platforms could use the model to learn about their customers and provide an improved level of service.

## A2. Objectives and Goals

To perform sentiment analysis, the review text must be cleaned, tokenized, and vectorized. The vectors must then be made the same length and split into training and testing data sets. The model will learn on the training set, then be applied to the testing set for accuracy review. The process will be outlined in more detail in a later section.

## A3. Prescribed Network

The chosen model for this analysis is *Sequential()* from the Tensorflow Keras package. *Sequential()* is a Recurrent Neural Network (RNN) which means it uses prior elements to determine its output. This helps it account for the position of the words in the reviews and gain a more complete understanding of the sentiment therein.

# Part II: Data Preparation

## B1. Data Exploration

The combined datasets contained a total of 17 reviews with one or more non-ascii characters. Once identified, the entire *review* column was restricted to characters with a Unicode value of less than 128 (the size of the ascii character list). This was further restricted to only upper/lowercase characters “a” through “z” then converted to all lowercase. Perhaps a more efficient sequence of steps exists; however, this one achieved its goal – there were no non-ascii characters remaining after this process.

Once the initial cleaning was completed, the size of the dictionary needed to be reduced. Therefore, the “stopwords” were removed (using the list of English stopwords in from *stopwords* the *nltk.corpus* package) and the remaining words were lemmatized (using the *nltk* package). The reviews were then converted into a single list of all words and subsequently turned into a *set* to reduce list to only unique words. There are 4766 unique words in the dataset “dictionary.”

The proposed embedding length is 32 which allows for a good level of complexity in the model while still running efficiently and accurately.

The maximum sequence length was originally set at the maximum review length of 686 words. This resulted in a first attempt at a model which was highly inaccurate (about as accurate as coin-flip at best). Further analysis on the sentence length revealed the majority of reviews with 10 or fewer words – a mean of 7, median of 5, and mode of 3. To improve the accuracy of the model, the sequence length was restricted to the mean (7) – this resulted in a model well over 90% accurate after only a few epochs of training.

Text

Description automatically generated Table

Description automatically generated

***Figure 1:*** *Analysis of Sentence Length (see code lines 63-90)*

## B2. Tokenization

In order for the model to learn word patterns, the words must be assigned a numerical “index” so that statistical analysis can be applied to the sentences. This process creates the dictionary. The function used was *Tokenizer* from the *keras.preprocessing.text* package.

Text

Description automatically generated



***Figure 2:*** *Tokenizing Code & Output*

Having the sentences tokenized into numerical values (and once they are all of equal length), the model is able to perform matrix operations to learn the patterns of the sentences.

## B3. Padding Process

The individual sentences are all padded to a consistent maximum length. In this instance, that length is 7 words (or, at this point, numbers). The initial attempt at building a model set the *maxlen* parameter in Figure 3 at the length of the longest review which was 686 words after pre-processing. After performing an analysis on the length of each review (see Figure 1), the *maxlen* parameter was changed to the *mean* length of each review (rounded to the nearest word). Because the longest review was an extreme outlier, the model performed very poorly. However, the model performed much better after reducing the length to a more relevant value.

Text

Description automatically generated

A picture containing text, electronics, keyboard, black

Description automatically generated

***Figure 3:*** *Padding Code and Output*

## B4. Categories of Sentiment

The dataset has only two categories of sentiment: 1 (positive) and 0 (negative). Therefore, the to the activation function of the final dense layer is *sigmoid*.

## B5. Steps to Prepare the Data

The steps thus far have been outlined in more detail in their respective sections. In summary, the data have been cleaned of non-ascii characters other than lowercase “a” through “z”, stopwords have been removed, and each word has been lemmatized. The reviews were then tokenized, converted to sequences, and padded to a length of 7.

The data are now ready to be split into training and testing datasets using the *train\_test\_split()* function in *sklearn.model\_selection*. The padded review vectors and ratings (converted into a corresponding list) were then passed into the *train\_test\_split()* function to create a training set with 70% of the data and a testing set with the remaining 30%. The training sets have 1331 reviews/ratings and the test sets have 571.

Text

Description automatically generated

Calendar

Description automatically generated Calendar

Description automatically generated

Calendar

Description automatically generatedCalendar

Description automatically generated

***Figure 4:*** *Train Test Split & Review*

## B6. Prepared Dataset

Please see the attached *ratings.csv* and *review\_vectors.csv* of the finalized data used in the model.

# Part III: Network Architecture

## C1. Model Summary

The code for the model and its output is provided below.

A screenshot of a computer

Description automatically generated

Text

Description automatically generated

***Figure 5:*** *Model Code and Summary*

## C2. Network Architecture

There were six layers used in this model. The first embedding layer, followed by a Spatial Dropout 1-dimensional layer, a Long Short-Term Memory layer, and finally two dense layers. After setting the embedding layer, the spatial dropout reduces overfitting by the LSTM layer. The dropout layer following LSTM provides the same reduction in overfitting. The final dense layer reduces the dimension to 1. There are 243,256 total parameters, all of which are trainable (meaning there are no non-trainable parameters).

## C3. Hyperparameters

The activation function used in the final dense layer is *sigmoid* because the output of the model needs to be either 1 (positive) or 0 (negative) to match the rating values. The embedding layer and spatial dropout layer both have

 Justify the choice of hyperparameters, including the following elements:

•   activation functions

•   number of nodes per layer

•   loss function

•   optimizer

•   stopping criteria

•   evaluation metric

# Part IV: Model Evaluation

## D1. Stopping Criteria

The

## D2. Training Process

The

## D3. Fit

The

## D4. Predictive Accuracy

The.

# Part V: Summary and Recommendations

## E. Code

please

## F. Functionality

See Figure 10 in section D3.

## G. Recommendations

The

# Part VI: Reporting

## H. Reporting

Please see attached code output.

## G. Sources of Third-Party Code

No third-party code was used in the execution of this script.

## H. Sources

**References**