D213 Task II - Sentiment Analysis using Neural Networks

David Harvell
Master of Science, Data Analytics
December 2021

A-1. Summarize one research question that you will answer using neural network models and NLP techniques. Be sure the research question is relevant to a real-world organizational situation and sentiment analysis captured in your chosen dataset.

We will be concerned with the following question during this analysis : **Can we determine the positive or negative sentiment of a review using Neural Networks?** We will be using review datasets from Amazon, IMDB, and Yelp.

A-2. Define the objectives or goals of the data analysis. Be sure the objectives or goals are reasonable within the scope of this research question and are represented in the available data.

If we are able to create a reliable model, **we can use Neural Networks to process millions of reviews and determine sentiment for our products.** This method would be exponential faster than having teams process the data, and it could deliver insights that previously had a high barrier of entry.

A-3. Identify a type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences from the selected data set.

We will use a standard **Bidirectional With (Artificial Neural Network) embedding Embedding for stronger word associations** with an end goal of binary classification. Preliminary testing was done using the Recurrent Network of LSTM, but the accuracy wasn't close enough to make confident predictions with the included datasets.

B-1. Perform exploratory data analysis on the chosen dataset, and include an explanation of each of the following elements/charts/graphs:

- presence of unusual characters (e.g., emojis, non-English characters, etc.)
- word clouds
- proposed word embedding length
- statistical justification for the chosen maximum sequence length

We begin by importing necessary libraries and importing the data.

```
In [1]: from pandas.ccbind import Series
        %time.matmul('/content/drive')
```

```
Mounted at /content/drive
```

```
In [2]: import pandas as pd
        import tensorflow as tf
        from tensorflow.keras.models import *
        from tensorflow.keras.layers
```

```
In [3]: # Create variable to point to the Google Drive
        path = '/content/drive/MyDrive/Datasets/'
        # the absolute path to your file
        filenames=list(.'amzn.txt', 'imdb.txt', 'yelp.txt')
        df = pd.DataFrame()
        # Read in all the files and concatenate
        for idx, file in enumerate(filenames):
            df2 = pd.read_csv(path + file, sep='\t', names=['Review', 'Sentiment'], header=None, quoting=3)
            df2['Review_Length'] = df2.Review.str.len()
            df2.fillna(0, inplace=True)
            df = pd.concat([df, df2])
        # Final look-see final dataframe and review info
        df.reset_index(drop=True)
        df.sample(frac=0.5).to_csv('amzn-imdb-yelp.csv', index=False)
        df.info()
```

```
<class> pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Review        3000 non-null   object
 1   Sentiment     3000 non-null   int64
 2   Review_Length 3000 non-null   int64
dtypes: int64(2), object(1)
memory usage: 70.3 KB
```

1. Non-alpha characters - We will cleanse the data to remove non-alpha values and excessive punctuations.

```
In [4]: # Data all words low and cover
        df.Review = df.Review.str.lower()

        # Remove punctuation in Review in words
        df.Review = df.Review.str.replace('[^\w\s]', '')
        df.sample(frac=0.5).head(4)
```

```
# Collect the length of review in words to see if we can keep later
df['Verificationlength'] = df.Review.str.len()

# Preview data
df.head()
```

Out[4]:

	Review	Sentiment	Review_Length	Review_Length
2	so there is no way for me to plug it in here in...	0	74	81
0	good case excellent value	1	26	26
2	great for the jawbone	1	21	21
2	fast charger for convenience taking more...	1	76	76
0	the mic is great	1	20	18

```
In [5]: # Create a list of the reviews and initialize variables for the vocabulary and max review length
        review_lines = df.Review.tolist()
        vocab = {}
        max_review_len = 0

        # Loop over the reviews, building our vocabulary and keeping track of the max review length
        for line in review_lines:
            review_len = len(review.split())
            if review_len > max_review_len:
                max_review_len = review_len

        for word in review.split():
            if word not in vocab:
                vocab[word] = len(vocab)

        print("Vocabs: ", len(vocab))
        print("Longest: ", max_review_len)
        print("Review length: ", len(review))
```

```
Vocabs:  5452
Longest:  2830
Review length:  4.3731149122326145
```

3. The vocabulary size is 5,452 distinct words.

1. Based on the vocabulary length, the suggested Embedding length is 8.9. We will move up to 10 to make sure sufficient vocabulary exist. This number is derived using the number of categories to the power of 1/4, based on suggestions from the Google developer blog (Google, 2017).

4. Our longest sentence uses 71 words, so that will be our target when padding the smaller values.

B-2. Describe the steps with the tokenization process, including any code generated and packages that are used to numerical text during the tokenization process.

https://developers.google.com/machine-learning/guides/text-classification/step-2-explore

We will use Keras built-in one-hot encoding to translate the words into numeric representations.

```
In [6]: review_reviews = [one_encoding(review, len(vocab), lower=True) for review in reviews]
        print(review_reviews[0])
```

```
[194, 3952, 2947, 76, 1402, 2447, 439, 476, 2194, 4442, 594, 2915, 594, 2391, 4194, 2970, 1206, 2029, 194, 2391, 527
```

B-3. Explain the padding process used to standardize the length of sequences, including the following in your explanation:

- if the padding occurs before or after the text sequence
- a screenshot of a single padded sequence

We are padding at the end of the reviews to make all sequences the same length as our max sequence (71 words). Below is the summarized result of the padding, followed by the entirety of the first review with padding.

```
In [7]: padded_reviews = keras.preprocessing.sequence.pad_sequences(encoded_reviews, maxlen=max_review_len, padding='post')
        print(padded_reviews)
```

```
[[ 194 3952 2947 ...   0   0   0]
 [ 194 3952 2947 ...   0   0   0]
 [ 194 3952 2947 ...   0   0   0]
 ...
 [1109 1208  422 ...   0   0   0]
 [2989 4194  439 ...   0   0   0]
 [3000 3872 2991 ...   0   0   0]]
```

```
In [8]: padded_reviews[0]
```

```
Out[8]: array([ 194, 3952, 2947,   76, 1402, 2447,  439,  476, 2194, 4442,  594,
        2915,  594, 2391, 4194, 2970, 1206, 2029,  194, 2391,  527,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0
```