Sentiment Analysis and Natural Language Processing

Charles Rowe, ID# 006539689, WGU D213 Advanced Data Analytics, Task 2

A1: Research Question

Using the provided texts, how accurately can an NLP model predict the sentiment of text.

A2: Objectives or Goals

The goals of this analysis are as follows:

- Concatenate provided texts into one dataframe
- Tokenize sentences
- Split unique words (minus stop words) into a bag-of-words
- Split data into training, testing, and validation sets
- Create sequential keras model for sentiment analysis
- Evaluate accuracy of model

A3: Prescribed Network

For this analysis, I will be using a Recurrent Neural Network (RNN). RNN networks are useful for language processing and speech recognition as it loops through sequential data to learn for better accuracy.

B1: Data Exploration

To begin, I will need to import the libraries that will be needed for the analysis, concatenate the provided datasets, and make sure that there are no null reviews or labels.

In [ ]:

```
import random
random.seed(1987)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
import string
import warnings
warnings.filterwarnings('ignore')
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Embedding, SimpleRNN
from tensorflow.keras.models import Sequential
from nltk import word_tokenize, sent_tokenize
import nltk.corpus
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
tf.config.run_functions_eagerly(True)
pd.set_option("display.max_columns", None)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\charlesrowe\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\charlesrowe\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\charlesrowe\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\charlesrowe\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [2]:

```
imdb_db_full = pd.read_csv('imdb_labelled.txt', sep='\t')
amazon_db_full = pd.read_csv('amazon_cells_labelled.txt', sep='\t')
yelp_db_full = pd.read_csv('yelp_labelled.txt', sep='\t')

full_db = pd.concat((imdb_db_full,amazon_db_full,yelp_db_full), ignore_index=True)
full_db.to_csv('full_db.csv')
n_cols = full_db.shape[1]

reviews = full_db['review']
labels = full_db['label']

print(reviews)
print(labels)
```

```
0       A very, very, very slow-moving, aimless movie ...
1       Not sure who was more lost - the flat characte...
2       Attempting artiness with black & white and cle...
3            Very little music or anything to speak of.
4       The best scene in the movie was when Gerardo i...
                              ...
2995    I think food should have flavor and texture an...
2996                             Appetite instantly gone.
2997    Overall I was not impressed and would not go b...
2998    The whole experience was underwhelming, and I ...
2999    Then, as if I hadn't wasted enough of my life ...
Name: review, Length: 3000, dtype: object
0       0
1       0
2       0
3       0
4       1
       ..
2995    0
2996    0
2997    0
2998    0
2999    0
Name: label, Length: 3000, dtype: int64
```

In [3]:

```python
review_text = np.asarray(reviews)
review_text = review_text[~pd.isnull(review_text)]
print(f"There are {len(review_text)} reviews.")
```

```
There are 3000 reviews.
```

In [4]:

```python
rev_labels = np.asarray(labels)
rev_labels = rev_labels[~pd.isnull(rev_labels)]
print(f"There are {len(rev_labels)} labels.")
```

```
There are 3000 labels.
```

As shown, there are no null reviews or labels. I will now check for presence of unusual characters.

In [5]:

```python
import demoji

emojis = [demoji.findall(review.lower()) for review in review_text]

print(emojis)
```

```
[{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}]
```

There are no emojis.

Non-English characters:

(Below uses code from Ednalyn C. De Dios-Copy2.ipynb), 2023 and William J. Townsend, 2023)

In [6]:

```
def isEnglish(c):
    try:
        c.encode(encoding='utf-8').decode('ascii')
    except UnicodeDecodeError:
        return False
    else:
        return True

non_english = []
for index, row in full_db.iterrows():
    for l in list(row['review']):
        for c in l:
            if isEnglish(c) == True:
                pass
            else:
                non_english.append(c)
```

```
non_english[:10]
```

Out[6]:

```
['\x96', 'é', '\x85', '\x96', 'é', 'é', 'å', '\x96', '\x97', '\x96']
```

There are non-english characters. I will now check for the vocabulary size.

In [7]:

```
import unicodedata
import re

def clean_sentence(sentence):
    stops = nltk.corpus.stopwords.words('english') + list(string.punctuation) + ['.
    negation = ['but', 'not', "don't", "aren't", "couldn't", "doesn't", "hadn't", "
                "isn't", "shouldn't", "wouldn't"]
    for word in negation:
        stops.remove(word)
    sentence = (unicodedata.normalize('NFKD', sentence)
        .encode('ascii', 'ignore')
        .decode('utf-8', 'ignore')
        .lower())
    words = re.sub(r'[^\w\s]', '', sentence).split()
    word_list = [word for word in words if word not in stops]
    return word_list

def get_words(df, column):
    return clean_sentence(''.join(str(df[column].tolist())))
```

In [8]:

```
all_words = get_words(full_db, 'review')
all_freq = pd.Series(all_words).value_counts()

vocab_size = len(all_freq)
print('Vocabulary size: ' + str(vocab_size))
```

```
Vocabulary size: 5279
```

Now to find the word embedding length and the maximum sequence length.

In [9]:

```
max_sequence_embedding = int(round(np.sqrt(np.sqrt(vocab_size)), 0))
print('Max sequence embedding: ' + str(max_sequence_embedding))
```

```
Max sequence embedding: 9
```

In [10]:

```
list_of_lens = []
for index, row in full_db.iterrows():
    list_of_lens.append(len(clean_sentence(row['review'])))
sequence_length = max(list_of_lens)
print('Sequence length: ' + str(sequence_length))
```

```
Sequence length: 41
```

Summary:

Regex and the demoji library were used to find and handle non-english characters.
Vocabulary size is 5279. This was done by the value_counts() method after handling non-english characters and removing stopwords.
The proposed word embedding length is 9, which was found by using the fourth root of the

vocabulary size. (As recommended by Google)
Maximum sequence length is 41. This was found by calculating lengths of sequences.

B2: Tokenization

Tokenization is the process of breaking down texts into individual words or characters. For this analysis, it would be better to tokenize into individual words. After tokenization is initiailized, we can run tokens through a list and iterate a bag-of-words to find the final vocabulary size to feed into the model.

In [11]:

```python
sentence_tokens = [sent_tokenize(review.lower()) for review in review_text]
print(f'Tokenized into {len(sentence_tokens)} elements.')
sentence_tokens[0]
```

```
Tokenized into 3000 elements.
```

Out[11]:

```
['a very, very, very slow-moving, aimless movie about a distressed, drifting young
man.']
```

In [12]:

```python
stopwords = nltk.corpus.stopwords.words('english') + list(string.punctuation) + ['.

word_tokens = []

for sentences in sentence_tokens:
    temporary_list = []

    if (len(sentences)) == 1:
        words_in_sentence = word_tokenize(sentences[0])
        for word in words_in_sentence:
            if word == "n't":
                word = "not"
            temporary_list.append(word)
    else:
        num_sentences = len(sentences)

        for i in range(0, num_sentences):
            words_in_sentence = word_tokenize(sentences[i])
            for word in words_in_sentence:
                if word == "n't":
                    word = "not"
                temporary_list.append(word)

    if len(temporary_list) == 0:
        temporary_list = ['empty']

    trimmed_words = [word for word in temporary_list if word not in stopwords]
    word_tokens.append(trimmed_words)
```

In [13]:

```python
i = 0
bag_of_words = []
max_review_length = 0

for words in word_tokens:
    if len(words) > max_review_length:
        max_review_length = len(words)
        print(f"Max length: {max_review_length} word tokens.")
    for word in words:
        bag_of_words.append(word)
    i += 1

unique_words = set(bag_of_words)
num_uniq_words = len(unique_words)
print(f"Words: {len(bag_of_words)} \nUnique Words: {num_uniq_words} \nMaximum numbe
```

```
Max length: 7 word tokens.
Max length: 8 word tokens.
Max length: 18 word tokens.
Max length: 22 word tokens.
Max length: 27 word tokens.
Max length: 39 word tokens.
Max length: 41 word tokens.
Words: 18518
Unique Words: 5204
Maximum number of words: 41
```

In [14]:

```
word_tokens[621]
```

Out[14]:

```
['mature',
 'subtle',
 'script',
 'suggests',
 'occasionally',
 'brings',
 'dramatic',
 'focus',
 'underlying',
 'tensions',
 'well',
 'served',
 'perfect',
 'performances',
 'apart',
 'odd',
 'inappropriate',
 'smiling',
 'keira',
 'knightley',
 'prone',
 'though',
 'perhaps',
 'direction',
 'time',
 'characters',
 'often',
 'mention']
```

In [15]:

```
row_list = []
i = 0

for i in range(0, len(word_tokens)):
    temp_dict = {}
    temp_string = ""
    for word in word_tokens[i]:
        temp_string = temp_string + word + " "
    temp_dict.update({"recommended" : rev_labels[i]})
    temp_dict.update({"review_text" : temp_string})
    row_list.append(temp_dict)
    i += 1

new_df = pd.DataFrame(row_list)
rows_to_drop = np.where(new_df['recommended'] == 'label')
rows_to_drop

new_df = new_df.drop([0,1001,2002])
```

In [16]:

```
X = new_df['review_text']
y = new_df['recommended']

X_train, X_remaining, y_train, y_remaining = train_test_split(X, y, train_size=.7,
X_valid, X_test, y_valid, y_test = train_test_split(X_remaining, y_remaining, test_
```

In [17]:

```
tokenizer = Tokenizer(num_words= 5000, lower= False)
tokenizer.fit_on_texts(X_train)
vocab_size = len(tokenizer.word_index) + 1
print(f"Vocab is {vocab_size} elements long.")

word_index = tokenizer.word_index
word_index
```

```
Vocab is 4116 elements long.
```

```
Out[17]: {'good': 1,
          'great': 2,
          'movie': 3,
          'phone': 4,
          'film': 5,
          'one': 6,
          'like': 7,
          'service': 8,
          'food': 9,
          'time': 10,
          'place': 11,
          'bad': 12,
          'really': 13,
          'would': 14,
          'well': 15,
          'ever': 16,
          'best': 17,
          'even': 18,
          'back': 19,
          'go': 20,
          'also': 21,
          "'ve": 22,
          'product': 23,
          'could': 24,
          "''": 25,
          'love': 26,
          'made': 27,
          'nice': 28,
          'work': 29,
          'quality': 30,
          'see': 31,
          'first': 32,
          'use': 33,
          'excellent': 34,
          'get': 35,
          "'m": 36,
          'sound': 37,
          'better': 38,
          'recommend': 39,
          'battery': 40,
          'much': 41,
          'never': 42,
          'works': 43,
          'way': 44,
          'headset': 45,
          'think': 46,
          'make': 47,
          '10': 48,
          'worst': 49,
          'got': 50,
          'acting': 51,
          'characters': 52,
          'everything': 53,
          'disappointed': 54,
          'still': 55,
          '2': 56,
          'say': 57,
          'movies': 58,
          'price': 59,
          'real': 60,
          'enough': 61,
          'thing': 62,
          'experience': 63,
          'minutes': 64,
          'right': 65,
          'terrible': 66,
          'case': 67,
          'ear': 68,
          'money': 69,
          'definitely': 70,
          'people': 71,
          'seen': 72,
          'pretty': 73,
          'every': 74,
          'two': 75,
          'piece': 76,
          'look': 77,
          'poor': 78,
          'lot': 79,
          'little': 80,
          'friendly': 81,
          'going': 82,
          'used': 83,
          'know': 84,
          'worth': 85,
          'amazing': 86,
          'delicious': 87,
          'waste': 88,
          'easy': 89,
          'years': 90,
          'watching': 91,
          'many': 92,
          'highly': 93,
          'horrible': 94,
          'nothing': 95,
          'take': 96,
          'screen': 97,
          'night': 98,
          'quite': 99,
          "'d": 100,
          'far': 101,
          '1': 102,
          'story': 103,
          'script': 104,
          'character': 105,
          'funny': 106,
          'actors': 107,
          'times': 108,
          'life': 109,
          'came': 110,
          'around': 111,
          'staff': 112,
          'overall': 113,
          'probably': 114,
          'scenes': 115,
          'went': 116,
          'give': 117,
          'music': 118,
          'however': 119,
          'new': 120,
          'day': 121,
          'camera': 122,
          'buy': 123,
          'anything': 124,
          'wonderful': 125,
          'must': 126,
          'item': 127,
          'bought': 128,
          'us': 129,
          'cast': 130,
          'comfortable': 131,
          'fine': 132,
          'last': 133,
          'anyone': 134,
          'plot': 135,
          'least': 136,
          'another': 137,
          'films': 138,
          'long': 139,
          'bluetooth': 140,
          'big': 141,
          'black': 142,
          '5': 143,
          'old': 144,
          'loved': 145,
          'want': 146,
          'car': 147,
          'ordered': 148,
          'show': 149,
          'always': 150,
          'come': 151,
          'eat': 152,
          'cheap': 153,
          'cool': 154,
          'though': 155,
          'thought': 156,
          'beautiful': 157,
          'fresh': 158,
          'watch': 159,
          'happy': 160,
          'kind': 161,
          'stars': 162,
          'worked': 163,
          'problems': 164,
          'simply': 165,
          'part': 166,
          'awesome': 167,
          'stupid': 168,
          'pizza': 169,
          'next': 170,
          'since': 171,
          'bit': 172,
          'talk': 173,
          'working': 174,
          'without': 175,
          'customer': 176,
          'fantastic': 177,
          'worse': 178,
          'awful': 179,
          'makes': 180,
          'done': 181,
          'small': 182,
          'hard': 183,
          'purchase': 184,
          'reception': 185,
          'restaurant': 186,
          'hear': 187,
          'menu': 188,
          'server': 189,
          'job': 190,
          'expect': 191,
          'atmosphere': 192,
          'actually': 193,
          'everyone': 194,
          'vegas': 195,
          'end': 196,
          'special': 197,
          'low': 198,
          'cell': 199,
          'three': 200,
          'totally': 201,
          'impressed': 202,
          'try': 203,
          'man': 204,
          'order': 205,
          'family': 206,
          'side': 207,
          'performance': 208,
          'tried': 209,
          'year': 210,
          'found': 211,
          'chicken': 212,
          'looks': 213,
          'said': 214,
          'felt': 215,
          'mess': 216,
          'enjoy': 217,
          'things': 218,
          'writing': 219,
          'gets': 220,
          'disappointing': 221,
          'sucks': 222,
          'fact': 223,
          'barely': 224,
          'return': 225,
          'believe': 226,
          'huge': 227,
          'completely': 228,
          'perfectly': 229,
          'started': 230,
          'especially': 231,
          'whole': 232,
          'seriously': 233,
          'charger': 234,
          'motorola': 235,
          'either': 236,
          'calls': 237,
          'line': 238,
          'unfortunately': 239,
          'absolutely': 240,
          'problem': 241,
          'bland': 242,
          'scene': 243,
          'bar': 244,
          'home': 245,
          'sucked': 246,
          'play': 247,
          'find': 248,
          'coming': 249,
          'white': 250,
          'took': 251,
          'call': 252,
          'series': 253,
          'volume': 254,
          'sushi': 255,
          'fast': 256,
          'left': 257,
          'short': 258,
          'fits': 259,
          'super': 260,
          'today': 261,
          'taste': 262,
          'different': 263,
          'original': 264,
          'feel': 265,
          'sure': 266,
          'watched': 267,
          'perfect': 268,
          'extremely': 269,
          'using': 270,
          'crap': 271,
          'design': 272,
          'enjoyed': 273,
          'interesting': 274,
          'salad': 275,
          'care': 276,
          'glad': 277,
          '3': 278,
          'boring': 279,
          'steak': 280,
          'kids': 281,
          'places': 282,
          'rather': 283,
          'days': 284,
          'meal': 285,
          'effects': 286,
          'put': 287,
          'away': 288,
          'fit': 289,
          'company': 290,
          'keep': 291,
          'top': 292,
          'phones': 293,
          'helpful': 294,
          'hot': 295,
          'tell': 296,
          'wait': 297,
          'getting': 298,
          'slow': 299,
          'second': 300,
          'gave': 301,
          'recommended': 302,
          'hour': 303,
          'mediocre': 304,
          'star': 305,
          'art': 306,
          'disappointment': 307,
          'device': 308,
          'lost': 309,
          'world': 310,
          'saw': 311,
          'plug': 312,
          'waited': 313,
          'charge': 314,
          'dishes': 315,
          'soon': 316,
          'something': 317,
          'area': 318,
          'broke': 319,
          'looking': 320,
          'dialogue': 321,
          'easily': 322,
          'picture': 323,
          'feeling': 324,
          'girl': 325,
          'comes': 326,
          'tv': 327,
          'mind': 328,
          'cinematography': 329,
          'flavor': 330,
          'may': 331,
          'none': 332,
          'software': 333,
          'john': 334,
          'ending': 335,
          'pleased': 336,
          'average': 337,
          'told': 338,
          'buffet': 339,
          'oh': 340,
          'point': 341,
          'spot': 342,
          'game': 343,
          'garbage': 344,
          'free': 345,
          'liked': 346,
          'almost': 347,
          'prices': 348,
          'months': 349,
          'directing': 350,
          'cable': 351,
          'self': 352,
          'tasty': 353,
          'selection': 354,
          'light': 355,
          'director': 356,
          'set': 357,
          'received': 358,
          'twice': 359,
          'predictable': 360,
          'house': 361,
          'burger': 362,
          'week': 363,
          'wasted': 364,
          'clear': 365,
          'ago': 366,
          'action': 367,
          'horror': 368,
          'close': 369,
          'i': 370,
          'hands': 371,
          'wear': 372,
          'wrong': 373,
          'beyond': 374,
          'inside': 375,
          'understand': 376,
          'large': 377,
          'simple': 378,
          'location': 379,
          'junk': 380,
          'decent': 381,
          'human': 382,
          'blue': 383,
          'kept': 384,
          'living': 385,
          'eating': 386,
          'amazon': 387,
          'store': 388,
          'together': 389,
          'fun': 390,
          'mostly': 391,
          'style': 392,
          'need': 393,
          '4': 394,
          'dish': 395,
          'stay': 396,
          'pictures': 397,
          'incredible': 398,
          'color': 399,
          'obviously': 400,
          'outside': 401,
          'course': 402,
          'hope': 403,
          'pay': 404,
          'hit': 405,
          'although': 406,
          'cinema': 407,
          'full': 408,
          'drama': 409,
          'friends': 410,
          'spicy': 411,
          'actor': 412,
          'internet': 413,
          'the': 414,
          'hours': 415,
          'deal': 416,
          'breakfast': 417,
          'overpriced': 418,
          'whatever': 419,
          'sandwich': 420,
          'verizon': 421,
          'meat': 422,
          'seems': 423,
          'goes': 424,
          'tasted': 425,
          'cold': 426,
          'certainly': 427,
          'feels': 428,
          'rude': 429,
          'yet': 430,
          '20': 431,
          'hate': 432,
          'waitress': 433,
          'truly': 434,
          'wanted': 435,
          'waiter': 436,
          'might': 437,
          'room': 438,
          'audio': 439,
          'turn': 440,
          'important': 441,
          'nokia': 442,
          'priced': 443,
          'fries': 444,
          'fails': 445,
          'superb': 446,
          'difficult': 447,
          'quick': 448,
          'guess': 449,
          'half': 450,
          'word': 451,
          'buttons': 452,
          'clean': 453,
          'needs': 454,
          'table': 455,
          'holes': 456,
          'trash': 457,
          'walked': 458,
          'written': 459,
          'sad': 460,
          'lacks': 461,
          'throughout': 462,
          '90': 463,
          'perhaps': 464,
          'bother': 465,
          'thin': 466,
          'let': 467,
          'annoying': 468,
          'lovely': 469,
          'amount': 470,
          'soundtrack': 471,
          'non': 472,
          'form': 473,
          'portrayal': 474,
          'making': 475,
          'jabra': 476,
          'due': 477,
          'waiting': 478,
          'drive': 479,
          'sat': 480,
          '8': 481,
          'solid': 482,
          'dining': 483,
          'headphones': 484,
          'tables': 485,
          'dirty': 486,
          'looked': 487,
          'choice': 488,
          'playing': 489,
          'etc': 490,
          'brilliant': 491,
          'happened': 492,
          'lots': 493,
          'poorly': 494,
          'cases': 495,
          'plus': 496,
          'given': 497,
          'wall': 498,
          'management': 499,
          'weeks': 500,
          'within': 501,
          'value': 502,
          'ask': 503,
          'party': 504,
          'unless': 505,
          'played': 506,
          'stuff': 507,
          'silent': 508,
          'less': 509,
          'front': 510,
          'others': 511,
          'features': 512,
          'owner': 513,
          'pho': 514,
          'despite': 515,
          'reasonable': 516,
          'sauce': 517,
          'loud': 518,
          'useless': 519,
          'leave': 520,
          'else': 521,
          'sick': 522,
          'seemed': 523,
          'particularly': 524,
          'hilarious': 525,
          'finally': 526,
          'shipping': 527,
          'high': 528,
          'minute': 529,
          'joy': 530,
          'sturdy': 531,
          'please': 532,
          'mistake': 533,
          'feature': 534,
          'couple': 535,
          'headsets': 536,
          'strong': 537,
          'budget': 538,
          'face': 539,
          'business': 540,
          'dropped': 541,
          'wind': 542,
          'damn': 543,
          'par': 544,
          'lunch': 545,
          'rare': 546,
          'trying': 547,
          'hitchcock': 548,
          'warm': 549,
          'maybe': 550,
          'served': 551,
          'chips': 552,
          'avoid': 553,
          'rent': 554,
          'voice': 555,
          'needed': 556,
          'hold': 557,
          'eyes': 558,
          'complete': 559,
          'station': 560,
          'pair': 561,
          'direction': 562,
          'expected': 563,
          'quickly': 564,
          'video': 565,
          'created': 566,
          'cant': 567,
          'intelligent': 568,
          '30': 569,
          'addition': 570,
          'songs': 571,
          'seeing': 572,
          'yes': 573,
          'stop': 574,
          'extra': 575,
          'easier': 576,
          'clarity': 577,
          'rice': 578,
          'bring': 579,
          'rating': 580,
          'seafood': 581,
          'space': 582,
          'ears': 583,
          'pathetic': 584,
          'main': 585,
          'imagination': 586,
          'indeed': 587,
          'sets': 588,
          'fried': 589,
          'shots': 590,
          'ready': 591,
          'mention': 592,
          'terrific': 593,
          'attentive': 594,
          'tacos': 595,
          'recently': 596,
          'failed': 597,
          'known': 598,
          'child': 599,
          'literally': 600,
          'death': 601,
          'gives': 602,
          'arrived': 603,
          'bargain': 604,
          'head': 605,
          'passed': 606,
          'bread': 607,
          'total': 608,
          'comedy': 609,
          'ended': 610,
          'exceptional': 611,
          'earlier': 612,
          'deserves': 613,
          'mouth': 614,
          'mobile': 615,
          'replace': 616,
          'wife': 617,
          'soup': 618,
          '7': 619,
          'classic': 620,
          'premise': 621,
          'asked': 622,
          'clever': 623,
          'ray': 624,
          'charles': 625,
          'serious': 626,
          'beer': 627,
          'razr': 628,
          'theater': 629,
          'tea': 630,
          'seem': 631,
          'connection': 632,
          'performances': 633,
          'water': 634,
          'weak': 635,
          'bacon': 636,
          'computer': 637,
          'cingular': 638,
          'network': 639,
          'dont': 640,
          'pasta': 641,
          'touch': 642,
          'authentic': 643,
          'lines': 644,
          'longer': 645,
          'buying': 646,
          'effective': 647,
          'killer': 648,
          'setting': 649,
          'leather': 650,
          'wine': 651,
          'list': 652,
          'ambiance': 653,
          'heart': 654,
          'fall': 655,
          'believable': 656,
          'share': 657,
          'dry': 658,
          'cost': 659,
          'plastic': 660,
          'review': 661,
          'utterly': 662,
          'unreliable': 663,
          'elsewhere': 664,
          'shrimp': 665,
          'consider': 666,
          'dinner': 667,
          'hand': 668,
          'ups': 669,
          'thriller': 670,
          'flick': 671,
          'range': 672,
          'offers': 673,
          'options': 674,
          'ten': 675,
          'lame': 676,
          'plain': 677,
          'says': 678,
          '12': 679,
          'eaten': 680,
          'sub': 681,
          'ringtones': 682,
          'reviews': 683,
          'particular': 684,
          'generally': 685,
          'tender': 686,
          'dead': 687,
          'sense': 688,
          'memorable': 689,
          'potato': 690,
          'loves': 691,
          'running': 692,
          'anytime': 693,
          'number': 694,
          'fairly': 695,
          'steaks': 696,
          'immediately': 697,
          'considering': 698,
          'fans': 699,
          'bored': 700,
          'check': 701,
          'whether': 702,
          'lacking': 703,
          'visit': 704,
          'crowd': 705,
          'embarrassing': 706,
          'attention': 707,
          'machine': 708,
          'phoenix': 709,
          'pros': 710,
          'homemade': 711,
          'gone': 712,
          'talented': 713,
          'and': 714,
          'everywhere': 715,
          'green': 716,
          'convincing': 717,
          'casting': 718,
          'tiny': 719,
          'vibe': 720,
          'james': 721,
          'returning': 722,
          'cartoon': 723,
          'pork': 724,
          'stage': 725,
          'overly': 726,
          'plays': 727,
          'credit': 728,
          'watchable': 729,
          'gem': 730,
          'grace': 731,
          'wooden': 732,
          'period': 733,
          'advise': 734,
          'provoking': 735,
          'significant': 736,
          'town': 737,
          'match': 738,
          'touching': 739,
          'unit': 740,
          'cooked': 741,
          'hell': 742,
          'comfort': 743,
          'fan': 744,
          'anne': 745,
          'sitting': 746,
          'results': 747,
          'racial': 748,
          'towards': 749,
          'impressive': 750,
          'note': 751,
          'pull': 752,
          'daughter': 753,
          'mic': 754,
          'joke': 755,
          'aerial': 756,
          'already': 757,
          'wow': 758,
          'boyfriend': 759,
          'beautifully': 760,
          'subtle': 761,
          'drops': 762,
          'letting': 763,
          'sending': 764,
          'previous': 765,
          'satisfied': 766,
          'warning': 767,
          'dog': 768,
          'possible': 769,
          'sashimi': 770,
          'issues': 771,
          'weird': 772,
          'busy': 773,
          'shot': 774,
          'conclusion': 775,
          'boot': 776,
          'treated': 777,
          'shows': 778,
          'clichés': 779,
          'rolls': 780,
          'effect': 781,
          'speed': 782,
          'scary': 783,
          'placed': 784,
          'lead': 785,
          'static': 786,
          'establishment': 787,
          'games': 788,
          'vegetarian': 789,
          'showed': 790,
          'conversations': 791,
          'portions': 792,
          'insult': 793,
          'intelligence': 794,
          'putting': 795,
          'nobody': 796,
          'speak': 797,
          'keeps': 798,
          'data': 799,
          'pleasant': 800,
          'driving': 801,
          'plan': 802,
          'forget': 803,
          'added': 804,
          'bill': 805,
          '6': 806,
          'tip': 807,
          'puppets': 808,
          'flaw': 809,
          'thats': 810,
          'charged': 811,
          'roles': 812,
          'regular': 813,
          'butter': 814,
          'final': 815,
          'instead': 816,
          'visual': 817,
          'interest': 818,
          'thoroughly': 819,
          'support': 820,
          'suspense': 821,
          'pg': 822,
          'modern': 823,
          'audience': 824,
          'storyline': 825,
          'sprint': 826,
          'wish': 827,
          'stayed': 828,
          'complain': 829,
          'mickey': 830,
          'crazy': 831,
          'breaking': 832,
          'mom': 833,
          'summary': 834,
          't': 835,
          'forced': 836,
          'selections': 837,
          'owned': 838,
          'pricing': 839,
          'laughable': 840,
          'cult': 841,
          'ridiculous': 842,
          'brought': 843,
          'possibly': 844,
          'score': 845,
          'strange': 846,
          'bore': 847,
          'husband': 848,
          'guy': 849,
          'red': 850,
          'interested': 851,
          'history': 852,
          'lightweight': 853,
          'player': 854,
          'spring': 855,
          'basically': 856,
          'gross': 857,
          'website': 858,
          'bakery': 859,
          'charm': 860,
          'bars': 861,
          'charging': 862,
          'smart': 863,
          'fare': 864,
          'hill': 865,
          'parents': 866,
          'dialog': 867,
          'bottom': 868,
          'nut': 869,
          'send': 870,
          'be': 871,
          'including': 872,
          'duck': 873,
          'greek': 874,
          'dressing': 875,
          'flavorful': 876,
          'production': 877,
          'mini': 878,
          'desserts': 879,
          'several': 880,
          'attempt': 881,
          'folks': 882,
          'rated': 883,
          'saying': 884,
          'beans': 885,
          'clearly': 886,
          'worthless': 887,
          'chinese': 888,
          'ok': 889,
          'soggy': 890,
          'tool': 891,
          'level': 892,
          'ruthless': 893,
          'ways': 894,
          'multiple': 895,
          'balance': 896,
          'cafe': 897,
          'replacement': 898,
          'step': 899,
          'race': 900,
          'forever': 901,
          'expensive': 902,
          'chargers': 903,
          'reading': 904,
          'exquisite': 905,
          'lacked': 906,
          'released': 907,
          '40': 908,
          'sounds': 909,
          'reasonably': 910,
          'drinks': 911,
          'tasteless': 912,
          'checked': 913,
          'complaints': 914,
          'provided': 915,
          'whatsoever': 916,
          'effort': 917,
          'indulgent': 918,
          'rate': 919,
          'giallo': 920,
          'thumbs': 921,
          'missed': 922,
          '100': 923,
          'included': 924,
          'person': 925,
          'idea': 926,
          'salsa': 927,
          'paid': 928,
          'beef': 929,
          'stand': 930,
          'q': 931,
          'holds': 932,
          'spend': 933,
          'lion': 934,
          'worthwhile': 935,
          'stories': 936,
          'unbelievable': 937,
          'palm': 938,
          'beep': 939,
          'signal': 940,
          'salmon': 941,
          'sides': 942,
          'dark': 943,
          'favorite': 944,
          'beat': 945,
          'mexican': 946,
          'delight': 947,
          'break': 948,
          'drop': 949,
          'thinking': 950,
          'takes': 951,
          'ability': 952,
          '15': 953,
          'sex': 954,
          'tom': 955,
          'adorable': 956,
          'ate': 957,
          'along': 958,
          'later': 959,
          'nasty': 960,
          'power': 961,
          'usb': 962,
          'seat': 963,
          'gon': 964,
          'na': 965,
          'example': 966,
          'supposedly': 967,
          'sweet': 968,
          'screenwriter': 969,
          'orders': 970,
          'trouble': 971,
          'incredibly': 972,
          'fish': 973,
          'exactly': 974,
          'earbud': 975,
          'checking': 976,
          'called': 977,
          'type': 978,
          'guys': 979,
          'son': 980,
          'actresses': 981,
          'scenery': 982,
          'belt': 983,
          'wireless': 984,
          'rocks': 985,
          'contained': 986,
          'opened': 987,
          'treat': 988,
          'documentary': 989,
          'wings': 990,
          'third': 991,
          'iphone': 992,
          'single': 993,
          'date': 994,
          'huston': 995,
          'able': 996,
          'five': 997,
          '50': 998,
          'stereotypes': 999,
          'appealing': 1000,
          ...}
```

Tokenization is complete. Vocab length of the training set is 4116 words.

B3: Padding Process

Padding is the process of adding new data (0s) to existing data to make it match the dimensions of the model. Since the maximum sequence length is 41, this is the number I will pad these sets to. Below is the code used to pad the "x" data within the training, testing, and validation sets.

In [18]:

```
X_train_seqs = tokenizer.texts_to_sequences(X_train)
X_train_pad = pad_sequences(X_train_seqs, maxlen=max_review_length, padding='post')
X_test_seq = tokenizer.texts_to_sequences(X_test)
X_test_pad = pad_sequences(X_test_seq, maxlen=max_review_length, padding='post')
X_val_seq = tokenizer.texts_to_sequences(X_valid)
X_val_pad = pad_sequences(X_val_seq, maxlen=max_review_length, padding='post')
```

In [19]:

```
print(f"The shape of the padded reviews within the training set is {X_train_pad.sha
print(X_train_pad)
```

```
The shape of the padded reviews within the training set is (2097, 41)
[[  91 1635  708 ...    0    0    0]
 [ 112   81 1014 ...    0    0    0]
 [   6  282  709 ...    0    0    0]
 ...
 [  26    2 4114 ...    0    0    0]
 [   8    1  290 ...    0    0    0]
 [   4  531 4115 ...    0    0    0]]
```

B4: Categories of Sentiment

For the model, two categories of sentiment will be used which references the labels of reviews (positive or negative). In the final layer, I have added the "sigmoid" activation that will transform the final values into either a 0 or 1.

B5: Steps to Prepare the Data

Summary of Steps:

- Concatenate all text files into one dataframe
- Initialize stopwords
- Clean data (remove special characters and stopwords)
- Tokenize words and find size of lexicon
- Split data into training, testing, and validation steps
- Pad "x" data

B6: Prepared Data Set

This will be included in the submission. Below is the code used to get the prepared data

In [20]:
```
x_train_df = pd.DataFrame(X_train_pad)
y_train_df = pd.DataFrame(y_train)
x_test_df = pd.DataFrame(X_test_pad)
y_test_df = pd.DataFrame(y_test)
x_val_df = pd.DataFrame(X_val_pad)
y_valid_df = pd.DataFrame(y_valid)

x_train_df.to_csv('x_train_df.csv')
y_train_df.to_csv('y_train.csv')
x_test_df.to_csv('x_test_pad.csv')
y_test_df.to_csv('y_test.csv')
x_val_df.to_csv('x_val_pad.csv')
y_valid_df.to_csv('y_valid.csv')
```

C1: Model Summary

In [22]:
```
model = Sequential()
model.add(SimpleRNN(41, input_shape=(32,1), activation='relu'))
model.add(Embedding(input_dim= vocab_size, output_dim= 8, input_length = max_review
model.add(Dense(150, activation = "relu"))
model.add(Dense(75, activation = "relu"))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer= 'adam', loss= 'binary_crossentropy', metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ simple_rnn (SimpleRNN)          │ (None, 41)             │         1,763 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ embedding (Embedding)           │ (None, 41, 8)          │        32,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 41, 150)        │         1,350 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 41, 75)         │        11,325 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 41, 1)          │            76 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 47,442 (185.32 KB)
 Trainable params: 47,442 (185.32 KB)
 Non-trainable params: 0 (0.00 B)
None
```

C2: Network Architecture

The final model consists of 5 layers (SimpleRNN, Embedding, and three Dense layers) with a final total of 47,442 parameters.

C3: Hyperparameters

- Activation functions: For the SimpleRNN and the first two Dense layers, I chose rectified linear unit (ReLU) as it is the industry standard. For the final Dense layer, I chose sigmoid as this sentiment analysis is being used on binary labels.
- Number of nodes per layer: There does not seem to be an industry standard for this hyperparameter. As such, I referred to Jason Brownlee who suggests analysts use "systemic experimentation".
- Loss function: Binary crossentropy was used because this is a binary classification.
- Optimizer: The optimizer used for this model is Adam (Adaptive Moment Estimation). Adam is a highly effective optimizer that is both memory-efficient and adaptive for each parameter.

D1: Stopping Criteria

Looking to the output of 23 below, our stopping criteria allowed the epochs to stop after 4 as the accuracy began to fall. This means that epoch 2 was seen as the highest accuracy and was shown to be the best, in accordance with the "patience" variable of 3.

In [23]:

```
num_classes = 41
y_train = to_categorical(y_train, num_classes=num_classes)
print(y_train.shape)
y_valid = to_categorical(y_valid, num_classes=num_classes)
print(y_valid.shape)
early_stop_check = EarlyStopping(monitor='val_accuracy', patience=3)
results = model.fit(X_train_pad, y_train, validation_data = (X_val_pad, y_valid), e
```

```
(2097, 41)
(450, 41)
Epoch 1/15
66/66 ━━━━━━━━━━━━━━━━━━━━ 6s 93ms/step - accuracy: 0.9076 - loss: 0.3853 - val_accu
racy: 0.9756 - val_loss: 0.1149
Epoch 2/15
66/66 ━━━━━━━━━━━━━━━━━━━━ 6s 94ms/step - accuracy: 0.9756 - loss: 0.1151 - val_accu
racy: 0.9756 - val_loss: 0.1147
Epoch 3/15
66/66 ━━━━━━━━━━━━━━━━━━━━ 6s 98ms/step - accuracy: 0.9756 - loss: 0.1148 - val_accu
racy: 0.9756 - val_loss: 0.1147
Epoch 4/15
66/66 ━━━━━━━━━━━━━━━━━━━━ 6s 96ms/step - accuracy: 0.9756 - loss: 0.1148 - val_accu
racy: 0.9756 - val_loss: 0.1147
```

D2: Fitness

Below shows the fitness of the model on the test set. It shows a loss of 0.115 and an accuracy of 0.976. This is a very high accuracy and shows that the model is a near perfect fit for the data.

As for overfitting, in previous versions of the model, I included a dropout metric. However, this caused more loss and increased the training time. For this model, I felt that the high accuracy and low loss in the model without dropout would be better so I scrapped the previous version.

In [24]:

```
y_test = to_categorical(y_test, num_classes=num_classes)
print(y_test.shape)
model.evaluate(X_test_pad, y_test)
```

```
(450, 41)
15/15 ━━━━━━━━━━━━━━━━━━━━ 1s 56ms/step - accuracy: 0.9756 - loss: 0.1147
```

Out[24]:

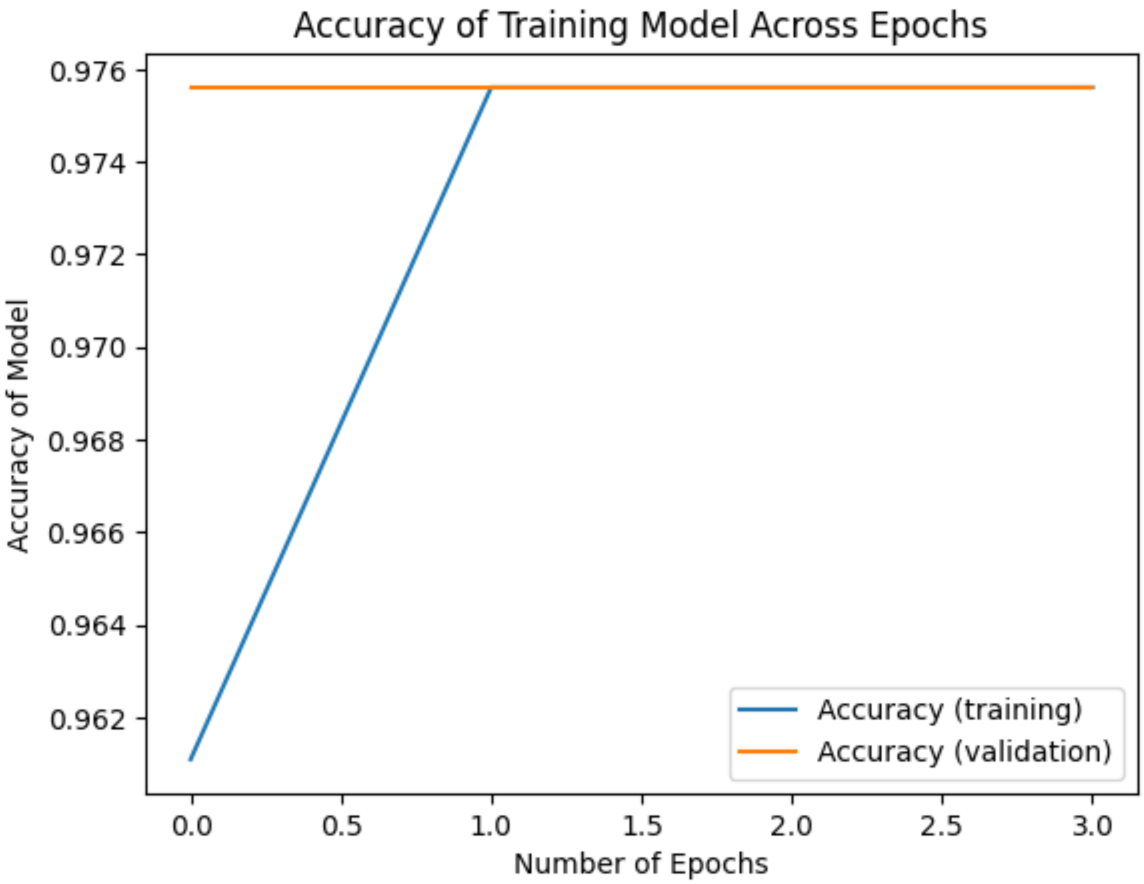
```
[0.11466538161039352, 0.9756097793579102]
```

D3: Training Process

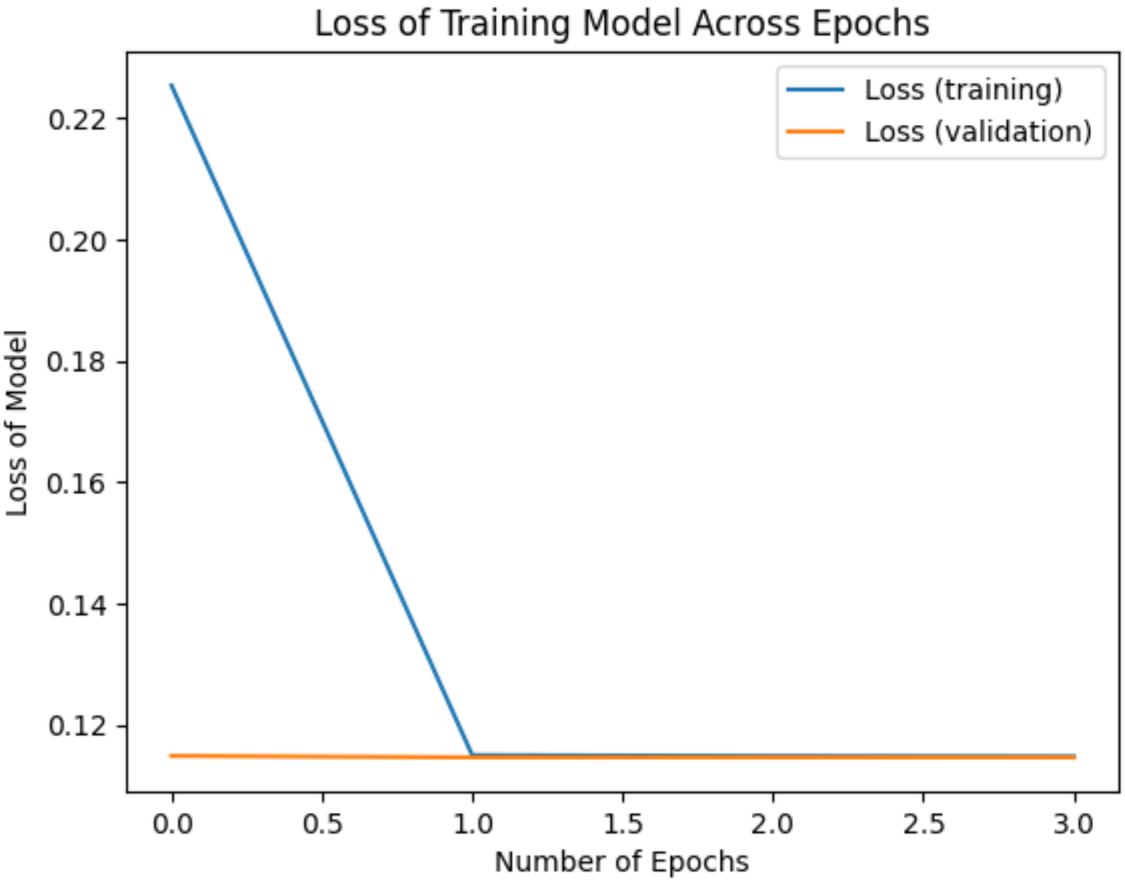
Below are visualizations of the model's training process, including the accuracy metric and loss.

In [25]:

```
plt.plot(results.history['accuracy'], label= "Accuracy (training)")
plt.plot(results.history['val_accuracy'], label= "Accuracy (validation)")
plt.xlabel("Number of Epochs")
plt.ylabel("Accuracy of Model")
plt.legend()
plt.title("Accuracy of Training Model Across Epochs")
plt.show()
```



In [26]:
```
plt.plot(results.history['loss'], label= "Loss (training)")
plt.plot(results.history['val_loss'], label= "Loss (validation)")
plt.xlabel("Number of Epochs")
plt.ylabel("Loss of Model")
plt.legend()
plt.title("Loss of Training Model Across Epochs")
plt.show()
```



D4: Predictive Accuracy

As shown above (24), the predictive accuracy of the model is 0.976 (97.6%) which is near perfect accuracy.

E: Trained Network Code

Below is the code to save the final model as a .keras file.

Please note: As this Jupyter notebook is saved on GitHub, the .keras file is too large to save. As such, I had to save it to Google Drive. It is available at: https://drive.google.com/drive/folders/1j08rk2cqKCgWuuEOjalbT_6eC9rFSXjb?usp=drive_link

In [27]:
```
import os
os.chdir('G:/My Drive/213 Files')
model.save('Task2Model.keras')

os.chdir('C:/Users/charlesrowe/Downloads/GitHub/WGU_MSDA/D213/Task 2')
```

F: Functionality

The final neural network is a near perfect fit for this data, showing an accuracy of 0.976 and a loss of 0.115. Due to these metrics, I believe this sentiment analysis was a resounding success.

In terms of network architecture, I believed it was fair to keep it simple for the first iteration of this model. As such, the model only consists of 5 layers. Though there is no exact way to determine the best network topology, since this is relatively simple data (consisting of just 6000 data points), I felt it would be better to keep it simple. A more complex model can be built later should a vast amount of data be added.

In relation, in adding dropout layers, more dense layers, and a flatten layer, it significantly increased execution time for no noticable increase in accuracy. This, however, did increase loss as well. As such, all of these extra layers were deemed unneeded and were not used in the final model.

G: Recommendations

At the current, my recommendations are as follows:

- While the model is very highly accurate, the model would most likely benefit from more data. Once more data is collected, the model can be compiled again to see if there is an increase in accuracy
- Future research will be needed to determine the continued accuaracy of this model. As language changes, words can shift their meanings with how they are used. It might be possible that this model will need to be updated in the future due to changes in word usage, slang, and public sentiment

H: Reporting

This notebook will be provided as a pdf in the submission and the repository

I: Sources of Third-Party Code

Ednalyn C. De Dios: https://github.com/ecdedios/d213-advanced-data-analytics/blob/main/task2/notebooks/D213%20Performance%20Assessment%20Task%202%20(Rev.%200)-Copy2.ipynb
William J. Townsend: https://github.com/WJTownsend/WGU_Portfolio/blob/main/D213/d213task2.ipynb

J: Sources

Google: https://developers.googleblog.com/en/introducing-tensorflow-feature-columns/
Jason Brownlee: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/