Tensorflow_Sentiment_Analysis

October 22, 2024

[1]:
```python
# setup for multiple outputs from single cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

[2]:
```python
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split, KFold
import re
from collections import Counter
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from wordcloud import WordCloud
import spacy
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

```
2024-04-01 01:08:52.588043: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: SSE4.1 SSE4.2 AVX, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
```

[3]:
```python
##########################################################################################
# IMPORT CSV FILE TO DATA FRAME␣
↪###########################################################
##########################################################################################


# NOTE: Before importing the csv to a data fame, it was helpful to open the csv␣
↪in excel and
# counvert all data to a single font and font size. This made it easier to␣
↪remove all punctuations
# by assuring that they are all of the same font
```

```python
# Load the dataset from CSV file into a DataFrame
df_original_reviews = pd.read_csv('IMDB_Dataset.csv')
# Display the DataFrame
df_original_reviews
```

[3]:

```
                                                  review sentiment
0      One of the other reviewers has mentioned that ...  positive
1      A wonderful little production. <br /><br />The...  positive
2      I thought this was a wonderful way to spend ti...  positive
3      Basically there's a family where a little boy ...  negative
4      Petter Mattei's "Love in the Time of Money" is...  positive
...                                                  ...       ...
49995  I thought this movie did a down right good job...  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di...  negative
49997  I am a Catholic taught in parochial elementary...  negative
49998  I'm going to have to disagree with the previou...  negative
49999  No one expects the Star Trek movies to be high...  negative

[50000 rows x 2 columns]
```

[4]:

```python
######################=============== DATA CLEANING␣
↪====================#################

###########################################################################################
# FORMAT DATA FRAME␣
↪######################################################################
###########################################################################################

# Convert sentiments from positive/negative to 1/0
# Define a dictionary mapping string labels to integer values
label_map = {'positive': 1, 'negative': 0}
# Convert the last column to integer values based on the mapping
df_original_reviews['sentiment'] = df_original_reviews['sentiment'].
↪map(label_map)
# Display the updated DataFrame
df_original_reviews
```

[4]:

```
                                                  review  sentiment
0      One of the other reviewers has mentioned that ...          1
1      A wonderful little production. <br /><br />The...          1
2      I thought this was a wonderful way to spend ti...          1
3      Basically there's a family where a little boy ...          0
4      Petter Mattei's "Love in the Time of Money" is...          1
...                                                  ...        ...
49995  I thought this movie did a down right good job...          1
49996  Bad plot, bad dialogue, bad acting, idiotic di...          0
```

```
49997  I am a Catholic taught in parochial elementary…          0
49998  I'm going to have to disagree with the previou…          0
49999  No one expects the Star Trek movies to be high…          0

[50000 rows x 2 columns]
```

[6]:
```
# Label index and reset index to begin at 1
# Rename index and start index at value of 1
df_original_reviews.index = range(1, len(df_original_reviews) + 1)
df_original_reviews.index.name = 'index'
# Display the updated DataFrame
df_original_reviews
```

[6]:
```
                                                   review  sentiment
index
1      One of the other reviewers has mentioned that …          1
2      A wonderful little production. <br /><br />The…          1
3      I thought this was a wonderful way to spend ti…          1
4      Basically there's a family where a little boy …          0
5      Petter Mattei's "Love in the Time of Money" is…          1
…                                                    …        …
49996  I thought this movie did a down right good job…          1
49997  Bad plot, bad dialogue, bad acting, idiotic di…          0
49998  I am a Catholic taught in parochial elementary…          0
49999  I'm going to have to disagree with the previou…          0
50000  No one expects the Star Trek movies to be high…          0

[50000 rows x 2 columns]
```

[7]:
```
# Capture original text reviews for latter display after classification and␣
 ↪confusion matrix
# Split the dataset into texts and labels
texts_original = df_original_reviews['review'].values
labels_original = df_original_reviews['sentiment'].values
df_all_reviews = df_original_reviews
```

[8]:
```
###########################################################################################
# REMOVE STOP WORDS AND UNWANTED CHARACTERS␣
 ↪################################################
###########################################################################################

# Remove unwanted HTML strings, "<br /><br />", and other unwanted characters
unwanted_HTML_string = ['<br /><br />',
                        '...', ':', '?', '.', ';',
                        '!', ')','(','"', "'s",
                       '-', '*', "'","n't", ',']
# Create a regex pattern to match any unwanted string
```

```
unwanted_pattern = '|'.join(map(re.escape, unwanted_HTML_string))
# Iterate over each row in the specified column and remove unwanted HTML␣
↪characters/strings
for index, row in df_all_reviews.iterrows():
    unclean_string = row['review']  # Extract text from the specified column

    # Remove unwanted characters/strings using regular expressions
    clean_string = re.sub(unwanted_pattern, '', unclean_string)

    # Update the DataFrame with the cleaned text
    df_all_reviews.at[index, 'review'] = clean_string

# Display the updated DataFrame
df_all_reviews.head(5)
df_all_reviews.tail(5)
```

[8]:

index	review	sentiment
1	One of the other reviewers has mentioned that …	1
2	A wonderful little production The filming tech…	1
3	I thought this was a wonderful way to spend ti…	1
4	Basically there a family where a little boy Ja…	0
5	Petter Mattei Love in the Time of Money is a v…	1

[8]:

index	review	sentiment
49996	I thought this movie did a down right good job…	1
49997	Bad plot bad dialogue bad acting idiotic direc…	0
49998	I am a Catholic taught in parochial elementary…	0
49999	Im going to have to disagree with the previous…	0
50000	No one expects the Star Trek movies to be high…	0

[9]:

```
# Remove stop words
# Get the English stopwords
stop_words = set(stopwords.words('english'))

# Iterate over each row in the specified column and remove stopwords
for index, row in df_all_reviews.iterrows():
    text = row['review']  # Extract text from the specified column

    # Tokenize the text into words
    words = nltk.word_tokenize(text)

    # Remove stopwords from the tokenized words
    filtered_words = [word for word in words if word.lower() not in stop_words]

    # Join the filtered words back into a single string and update the DataFrame
```

```
    df_all_reviews.at[index, 'review'] = ' '.join(filtered_words)

# Display the updated DataFrame
df_all_reviews.head(5)
df_all_reviews.tail(5)
```

[9]:

index	review	sentiment
1	One reviewers mentioned watching 1 Oz episode …	1
2	wonderful little production filming technique …	1
3	thought wonderful way spend time hot summer we…	1
4	Basically family little boy Jake thinks zombie…	0
5	Petter Mattei Love Time Money visually stunnin…	1

[9]:

index	review	sentiment
49996	thought movie right good job creative original…	1
49997	Bad plot bad dialogue bad acting idiotic direc…	0
49998	Catholic taught parochial elementary schools n…	0
49999	Im going disagree previous comment side Maltin…	0
50000	one expects Star Trek movies high art fans exp…	0

[10]:

```
###########################################################################################
# BREAK MAIN DATA FRAME INTO POSITIVE AND NEGATIVE REWIEWS␣
  ↪################################
###########################################################################################

# Select negative review strings
df_negative_reviews = df_all_reviews[df_all_reviews['sentiment'] == 0]
print(len(df_negative_reviews))
```

25000

[11]:

```
# Select positive review strings
df_positive_reviews = df_all_reviews[df_all_reviews['sentiment'] == 1]
print(len(df_positive_reviews))
```

25000

[12]:

```
##############===============EXPLORATORY DATA ANALYSIS␣
  ↪=====================###############

###########################################################################################
# DISPLAY BAR CHARTS SHOWING TOP 25 WORDS FOR POSITIVE AND NEGATIVE REWIEWS␣
  ↪###############
###########################################################################################

# A function to display top 25 words in horizontal bar chart
```

```python
def top_25_bar(target_dataframe, sentiment, word_type):
    # extract strings in review column of dataframe
    strings = target_dataframe['review']

    # Join all strings in the column into a single string
    all_joined_strings = ' '.join(strings)

    # Split the text into words and count the frequency of each word
    word_counts = Counter(all_joined_strings.split())

    # Get the top 25 most common words
    top_words = dict(word_counts.most_common(25))

    # Create a horizontal bar chart
    plt.figure(figsize=(10, 8))
    plt.barh(list(top_words.keys()), list(top_words.values()), color='skyblue')
    plt.xlabel('Frequency')
    plt.ylabel('Word')
    title = 'Top 25 ' + str(sentiment) + ' ' + str(word_type) + ' in ' +␣
 ↪str(sentiment) + ' reviews'
    plt.title(title)
    plt.gca().invert_yaxis()  # Invert y-axis to have the most frequent word at␣
 ↪the top
    plt.show();

    return top_words, title
```

[13]:

```python
# Top 25 words in negative reviews
top_25_negative_all_words, title = top_25_bar(df_negative_reviews, 'negative',␣
 ↪'words')
count = 1
print("FIGURE ", count, "(above): ", title)
# Top 25 words in positive reviews
top_25_positive_all_words, title = top_25_bar(df_positive_reviews, 'positive',␣
 ↪'words')
count += 1
print("FIGURE ", count, "(above): ", title)
```

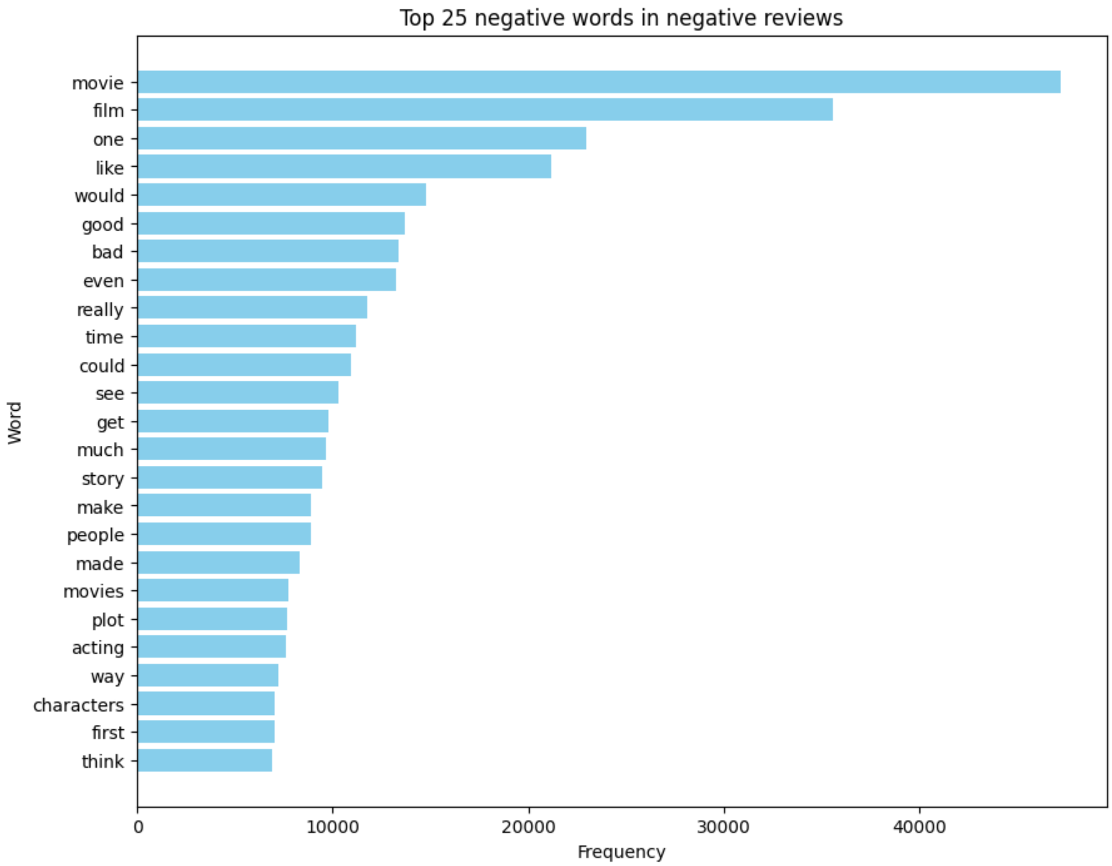


FIGURE 1 (above): Top 25 negative words in negative reviews

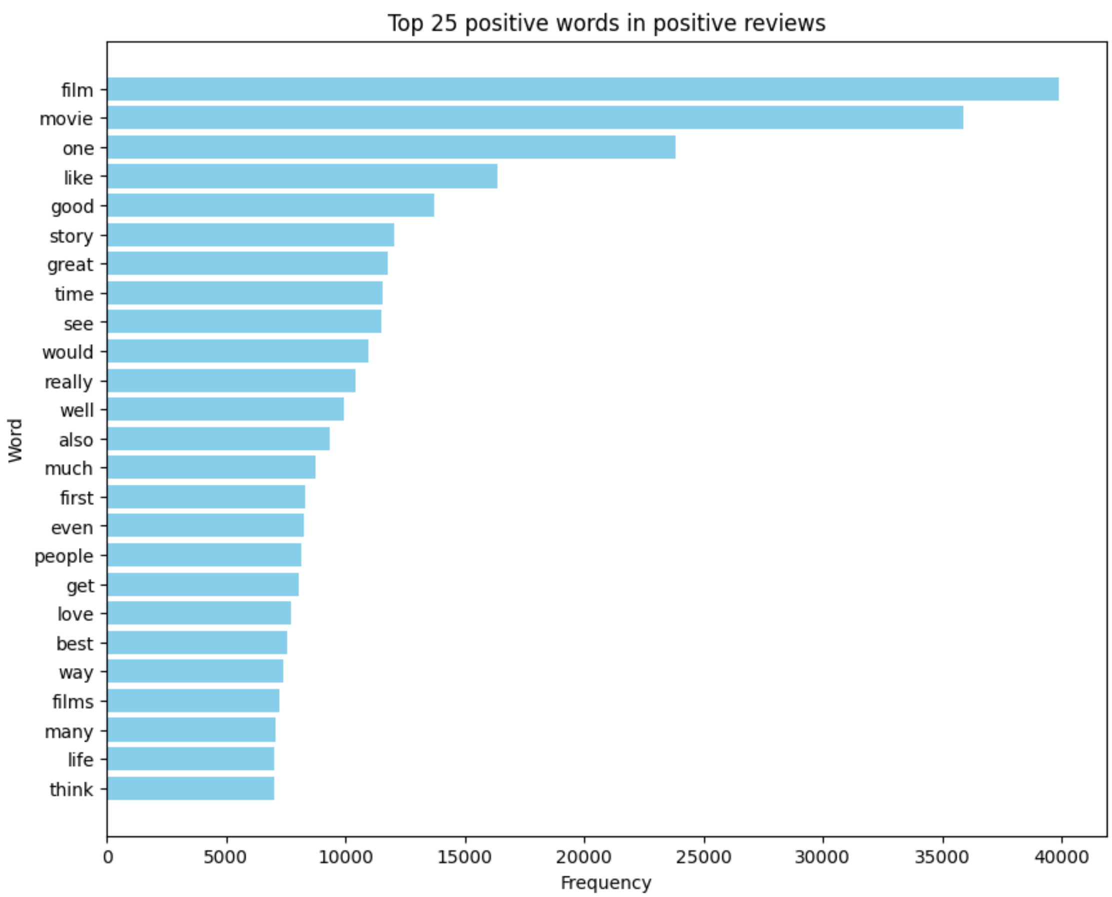


FIGURE 2 (above): Top 25 positive words in positive reviews

Observations: – The words 'good' and 'bad' appear at nearly equal word rank in negative reviews (figure 1). – The word 'good' appears without the word 'bad' much more often in positive reviews (figure 2). – The words 'good', 'great', 'best' and 'love' appear more often in positive reviews than in negative reviews (figure 2). – The words 'characters', 'story', and 'acting' appear more often in negative reviews (figure 1).

[14]:
```
##########################################################################################
# DISPLAY WORD CLOUDS SHOWING TOP 25 WORDS FOR POSITIVE AND NEGATIVE REWIEWS␣
 ↪##############
##########################################################################################

# A function to display top 25 words in word cloud
def top_25_wordcloud(target_dataframe, sentiment, word_type):
    # extract strings in review column of dataframe
    all_joined_strings = ' '.join(target_dataframe['review'])

    # Word cloud of top 100 words in all negative reviews
    word_cloud_parameters ={'width':800, 'height':400,
```

```
                                'background_color':'white',
                                'colormap':'viridis',
                                'max_words':100
                               }
    # Construct word cloud
    wordcloud = WordCloud(**word_cloud_parameters).generate(all_joined_strings)
    # Plot the word cloud
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    title = 'TOP 25 ' + str(sentiment) + ' ' + str(word_type) + ' IN ' +␣
↪str(sentiment) + ' REVIEWS'
    plt.title(title)
    plt.axis('off')
    plt.tight_layout()
    plt.show();

    return title
```

[15]:

```
# Top 25 words word cloud in negative reviews
title = top_25_wordcloud(df_negative_reviews, 'NEGATIVE', 'WORDS\n')
count += 1
print("FIGURE ", count, "(above): ", title)
# Top 25 words word cloud in positive reviews
title =  top_25_wordcloud(df_positive_reviews, 'POSITIVE', 'WORDS\n')
count += 1
print("FIGURE ", count, "(above): ", title)
```

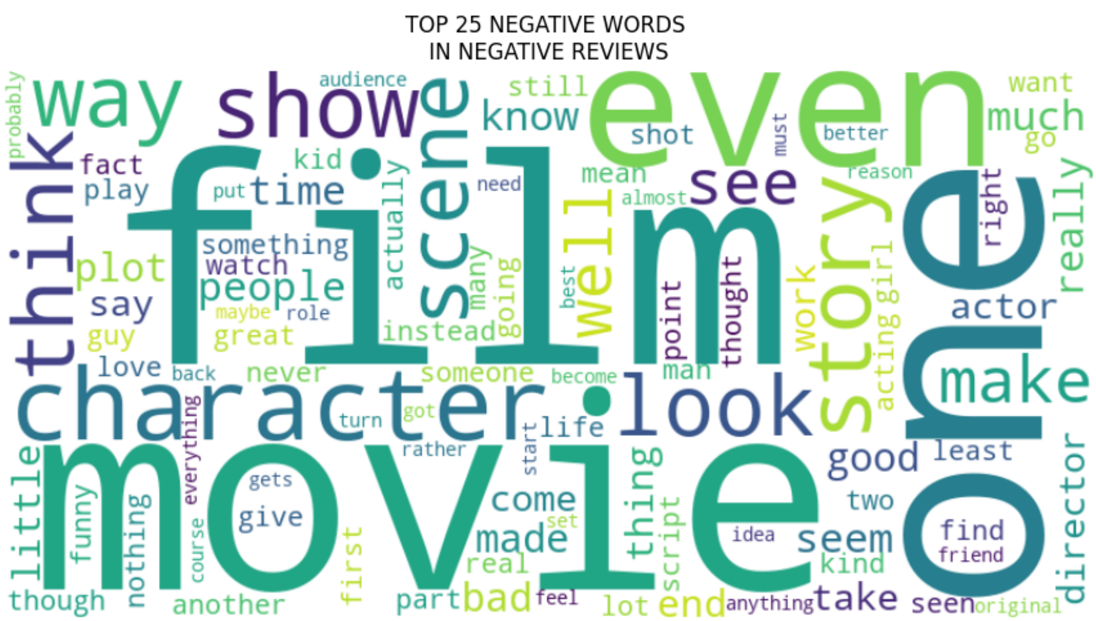


FIGURE 3 (above): TOP 25 NEGATIVE WORDS
 IN NEGATIVE REVIEWS

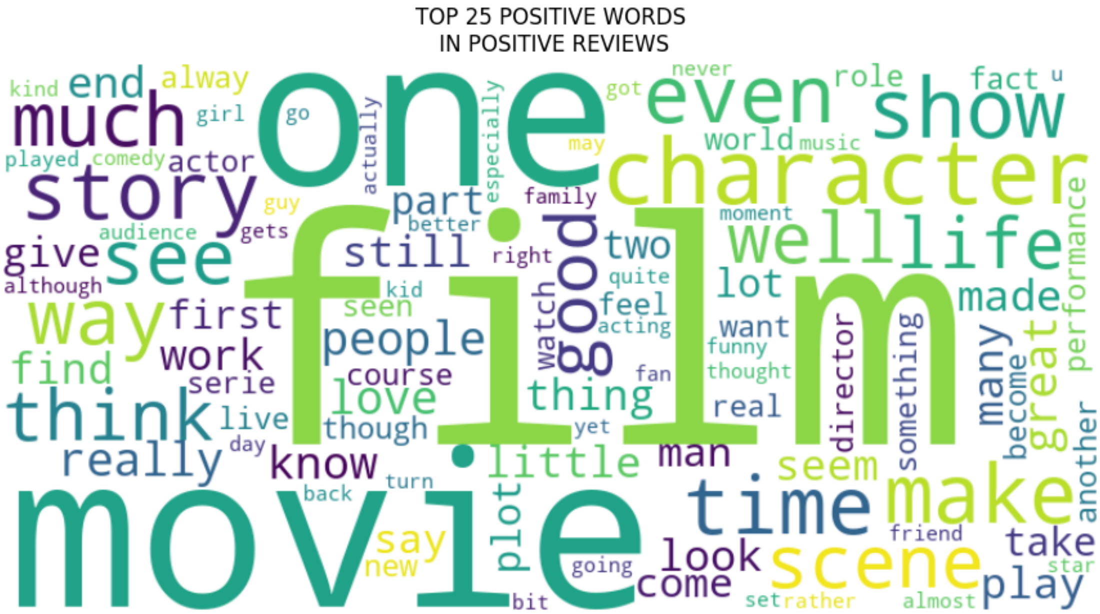


FIGURE 4 (above): TOP 25 POSITIVE WORDS
 IN POSITIVE REVIEWS

[16]:
```
##############################################################################################
# INSTALL SPACY FOR ADJECTIVE AND VERB DETECTION IN POSITIVE AND NEGATIVE␣
↪REWIEWS #########
##############################################################################################

# Extract 25 most common positive and negative adjectives in reviews
# May need to install spacy for next steps
!pip install spacy
```

```
Requirement already satisfied: spacy in ./anaconda3/lib/python3.10/site-packages
(3.7.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
./anaconda3/lib/python3.10/site-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
./anaconda3/lib/python3.10/site-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
./anaconda3/lib/python3.10/site-packages (from spacy) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
./anaconda3/lib/python3.10/site-packages (from spacy) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
./anaconda3/lib/python3.10/site-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in
```

```
./anaconda3/lib/python3.10/site-packages (from spacy) (8.2.3)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
./anaconda3/lib/python3.10/site-packages (from spacy) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
./anaconda3/lib/python3.10/site-packages (from spacy) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
./anaconda3/lib/python3.10/site-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in
./anaconda3/lib/python3.10/site-packages (from spacy) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in
./anaconda3/lib/python3.10/site-packages (from spacy) (0.9.4)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in
./anaconda3/lib/python3.10/site-packages (from spacy) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
./anaconda3/lib/python3.10/site-packages (from spacy) (4.66.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
./anaconda3/lib/python3.10/site-packages (from spacy) (2.29.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
./anaconda3/lib/python3.10/site-packages (from spacy) (1.10.12)
Requirement already satisfied: jinja2 in ./anaconda3/lib/python3.10/site-
packages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in ./anaconda3/lib/python3.10/site-
packages (from spacy) (65.6.3)
Requirement already satisfied: packaging>=20.0 in ./.local/lib/python3.10/site-
packages (from spacy) (23.0)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in
./anaconda3/lib/python3.10/site-packages (from spacy) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in ./.local/lib/python3.10/site-
packages (from spacy) (1.24.1)
Requirement already satisfied: typing-extensions>=4.2.0 in
./anaconda3/lib/python3.10/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
./anaconda3/lib/python3.10/site-packages (from requests<3.0.0,>=2.13.0->spacy)
(2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./anaconda3/lib/python3.10/site-
packages (from requests<3.0.0,>=2.13.0->spacy) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
./anaconda3/lib/python3.10/site-packages (from requests<3.0.0,>=2.13.0->spacy)
(1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in
./.local/lib/python3.10/site-packages (from requests<3.0.0,>=2.13.0->spacy)
(2022.12.7)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in
./anaconda3/lib/python3.10/site-packages (from thinc<8.3.0,>=8.2.2->spacy)
(0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
./anaconda3/lib/python3.10/site-packages (from thinc<8.3.0,>=8.2.2->spacy)
```

```
(0.1.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in
./anaconda3/lib/python3.10/site-packages (from typer<0.10.0,>=0.3.0->spacy)
(8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in
./anaconda3/lib/python3.10/site-packages (from weasel<0.4.0,>=0.1.0->spacy)
(0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
./anaconda3/lib/python3.10/site-packages (from jinja2->spacy) (2.1.3)

[notice] A new release of pip is
available: 23.3.2 -> 24.0
[notice] To update, run:
pip install --upgrade pip
```

[17]:
```
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-
any.whl (12.8 MB)

12.8/12.8 MB 2.7 MB/s eta 0:00:00m eta
0:00:01[36m0:00:01
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in
./anaconda3/lib/python3.10/site-packages (from en-core-web-sm==3.7.1) (3.7.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (8.2.3)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
```

```
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.9.4)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (4.66.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.29.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.10.12)
Requirement already satisfied: jinja2 in ./anaconda3/lib/python3.10/site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.1.2)
Requirement already satisfied: setuptools in ./anaconda3/lib/python3.10/site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (65.6.3)
Requirement already satisfied: packaging>=20.0 in ./.local/lib/python3.10/site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (23.0)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in
./anaconda3/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in ./.local/lib/python3.10/site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.24.1)
Requirement already satisfied: typing-extensions>=4.2.0 in
./anaconda3/lib/python3.10/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (4.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
./anaconda3/lib/python3.10/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./anaconda3/lib/python3.10/site-
packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
./anaconda3/lib/python3.10/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in
```

```
./.local/lib/python3.10/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2022.12.7)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in
./anaconda3/lib/python3.10/site-packages (from
thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
./anaconda3/lib/python3.10/site-packages (from
thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.1.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in
./anaconda3/lib/python3.10/site-packages (from
typer<0.10.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in
./anaconda3/lib/python3.10/site-packages (from
weasel<0.4.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
./anaconda3/lib/python3.10/site-packages (from jinja2->spacy<3.8.0,>=3.7.2->en-
core-web-sm==3.7.1) (2.1.3)

[notice] A new release of pip is
available: 23.3.2 -> 24.0
[notice] To update, run:
pip install --upgrade pip
 Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

[18]:
```python
###########################################################################################
# FIND ADJECTIVES IN POSITIVE AND NEGATIVE REWIEWS␣
 ↪########################################
###########################################################################################

# A function to extract ADJECTIVES in a group of sentiment reviews
def review_adjectives(target_dataframe):
    # Load spaCy English language model
    nlp = spacy.load("en_core_web_sm")

    # Iterate over each row in the specified column isolate adjectives
    for index, row in target_dataframe.iterrows():
        # Extract original individual string from the review column
        text = row['review']

        # Process the string with spaCy
        doc = nlp(text)

        # Extract adjectives from the string
        adjectives = [] # Define/reset empty array to hold extracted adjectives
        adjectives = [token.text for token in doc if token.pos_ == "ADJ"]
```

```
        # Join the adjectives back into a single adjective string
        # Insert adjective string into the DataFrame to replace original string
        target_dataframe.at[index, 'review'] = ' '.join(adjectives)

    return target_dataframe
```

[19]:
```
# Extract adjectives from negative reviews
negative_adjectives = review_adjectives(df_negative_reviews)
```

[20]:
```
negative_adjectives.head(5)
negative_adjectives.tail(5)
```

[20]:

index	review	sentiment
4	little slower watchable real similar meaningless	0
8	amazing fresh innovative brilliant funny compl…	0
9	positive Bad awful less cheap nasty boring hap…	0
11	quirky oddness actual odd funny oddness funny …	0
12	scariest big right young tired beautiful happy…	0

[20]:

index	review	sentiment
49995	typical genuine memorable clever bad ugly bori…	0
49997	Bad bad bad idiotic annoying crappy bad better…	0
49998	Catholic parochial elementary high good Cathol…	0
49999	disagree previous second vicious Western centr…	0
50000	high good best implausible worst far goofy wor…	0

[21]:
```
# Extract adjectives from positive reviews
positive_adjectives = review_adjectives(df_positive_reviews)
```

[22]:
```
positive_adjectives.head(5)
positive_adjectives.tail(5)
```

[22]:

index	review	sentiment
1	first right hearted timid classic experimental…	1
2	wonderful little entire seamless worth masterf…	1
3	wonderful hot lighthearted simplistic witty li…	1
5	stunning vivid different present different nex…	1
6	favorite noble preachy old last sympathetic de…	1

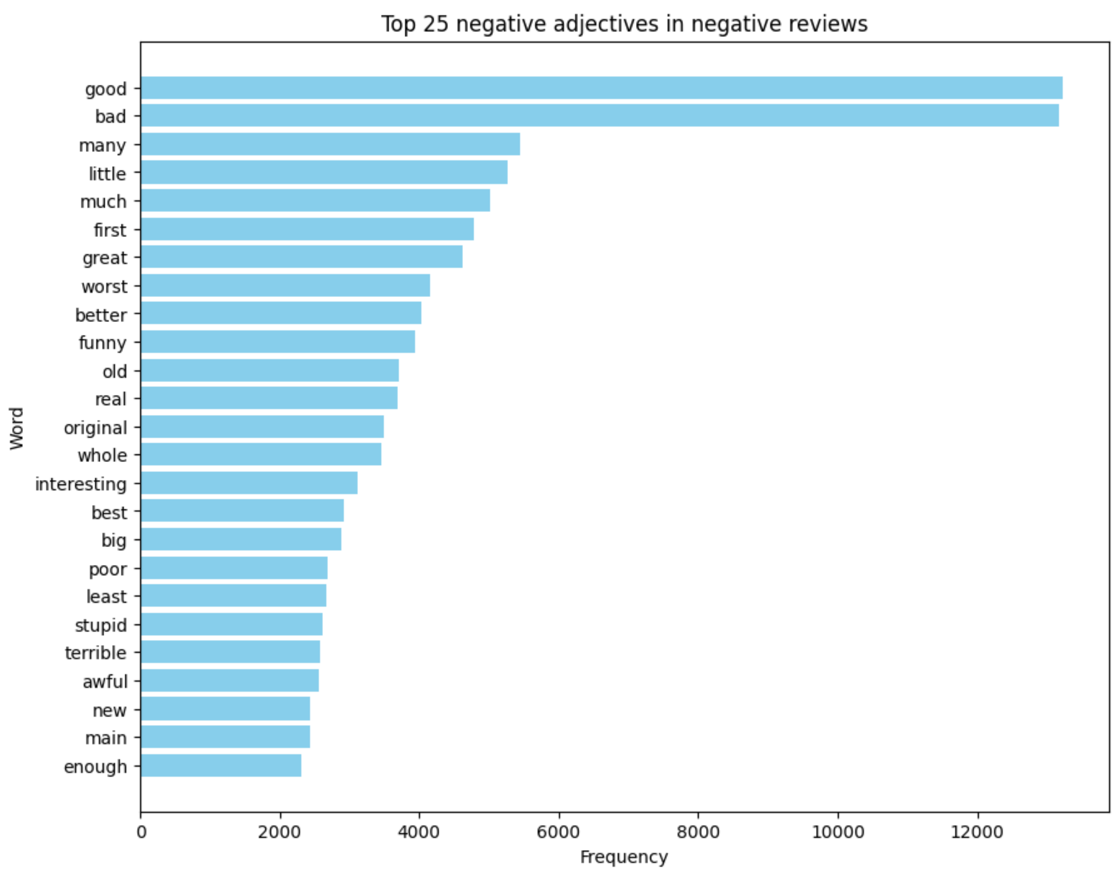
[22]:

index	review	sentiment
49984	original back whole new excellent special supe…	1
49986	best complete utter whole perfect much younger…	1
49990	modern true favorite Jewish top pure superfici…	1
49993	live stereotypical sure complex great present …	1

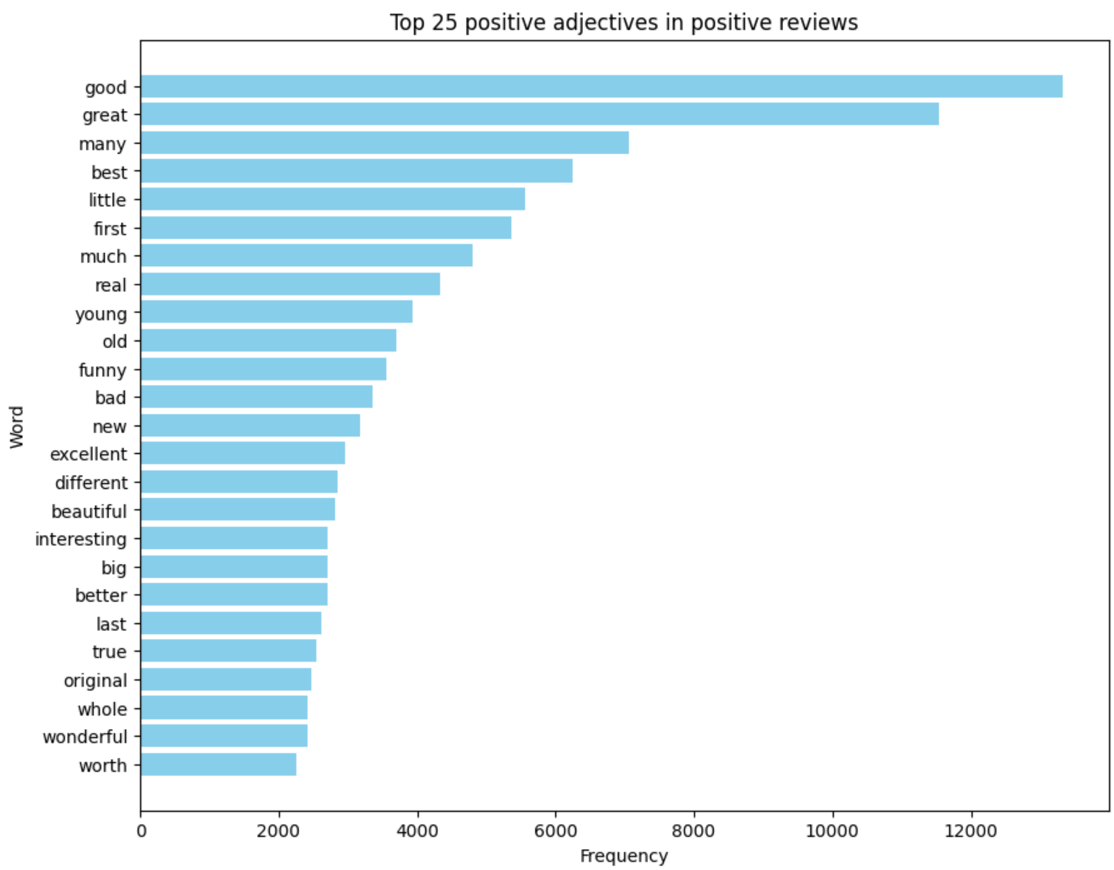
```
49996  good creative original first whole last underr…          1
```

[23]:
```
##########################################################################################
# DISPLAY BAR CHARTS SHOWING TOP 25 ADJECTIVES FOR POSITIVE AND NEGATIVE␣
 ↪REWIEWS ##########
##########################################################################################

# Top 25 adjectives in negative reviews
top_25_negative_adjective_words, title = top_25_bar(negative_adjectives,␣
 ↪'negative', 'adjectives')
count += 1
print("FIGURE ", count, "(above): ", title)
# Top 25 adjectives in positive reviews
top_25_positive_adjective_words, title = top_25_bar(positive_adjectives,␣
 ↪'positive', 'adjectives')
count += 1
print("FIGURE ", count, "(above): ", title)
```




```
FIGURE  5 (above):  Top 25 negative adjectives in negative reviews
```




```
FIGURE  6 (above):  Top 25 positive adjectives in positive reviews
```

[24]:
```
##########################################################################################
# DISPLAY WORD CLOUDS SHOWING TOP 25 ADJECTIVES FOR POSITIVE AND NEGATIVE␣
↪REWIEWS #########
##########################################################################################

# Top 25 adjectives word cloud in negative reviews
title = top_25_wordcloud(negative_adjectives, 'NEGATIVE', 'ADJECTIVES\n')
count += 1
print("FIGURE ", count, "(above): ", title)
# Top 25 adjectives word cloud in positive reviews
title = top_25_wordcloud(positive_adjectives, 'POSITIVE', 'ADJECTIVES\n')
count += 1
print("FIGURE ", count, "(above): ", title)
```

FIGURE 7 (above): TOP 25 NEGATIVE ADJECTIVES
IN NEGATIVE REVIEWS

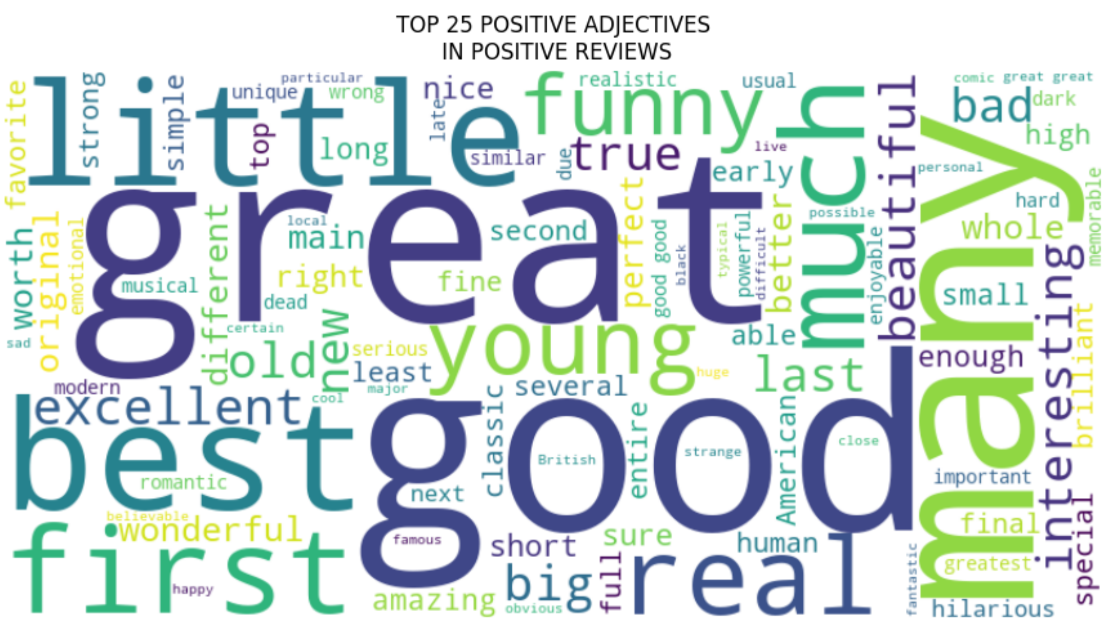


FIGURE 8 (above): TOP 25 POSITIVE ADJECTIVES
IN POSITIVE REVIEWS

Observations: – The adjectives 'good' and 'bad' appear at nearly equal word rank in negative reviews (figures 5 and 7). – The adjective 'good' appears without the word 'bad' much more often in positive reviews (figure 6). – The adjectives 'good', 'great', 'many', and 'best' appear more often in positive reviews than in negative reviews (figure 8).

[25]:
```
#############################################################################################
# FIND VERBS IN POSITIVE AND NEGATIVE REWIEWS␣
↪#############################################
#############################################################################################

# A function to extract VERBS in a group of sentiment reviews
def review_verbs(target_dataframe):
    # Load spaCy English language model
    nlp = spacy.load("en_core_web_sm")

    # Iterate over each row in the specified column isolate verbss
    for index, row in target_dataframe.iterrows():
        # Extract original individual string from the review column
        text = row['review']

        # Process the string with spaCy
        doc = nlp(text)

        # Extract verbss from the string
        verbs = [] # Define/reset empty array to hold extracted verbs
        verbs = [token.text for token in doc if token.pos_ == "VERB"]

        # Join the verbs back into a single verb string
        # Insert verb string into the DataFrame to replace original string
        target_dataframe.at[index, 'review'] = ' '.join(verbs)

    return target_dataframe
```

[26]:
```
# Extract verbs from negative reviews
negative_verbs = review_verbs(df_negative_reviews)
negative_verbs.head(5)
negative_verbs.tail(5)
```

[26]:

index	review	sentiment
4		0
8	fallen entertaining	0
9		0
11		0
12		0

[26]:

index	review	sentiment
49995	boring	0

```
49997                             0
49998          organized          0
49999  disagree mumbling          0
50000                             0
```

[27]:
```
# Extract verbs from positive reviews
positive_verbs = review_verbs(df_positive_reviews)
positive_verbs.head(5)
positive_verbs.tail(5)
```

[27]:
```
      review  sentiment
index
1                     1
2                     1
3                     1
5                     1
6                     1
```

[27]:
```
      review  sentiment
index
49984                 1
49986                 1
49990                 1
49993   live          1
49996                 1
```

[28]:
```
###########################################################################################
# DISPLAY BAR CHARTS SHOWING TOP 25 VERBS FOR POSITIVE AND NEGATIVE REWIEWS␣
 ↪###############
###########################################################################################

# Top 25 verbs in negative reviews
top_25_negative_verb_words, title = top_25_bar(negative_verbs, 'negative',␣
 ↪'verbs')
count += 1
print("FIGURE ", count, "(above): ", title)
# Top 25 verbs in positive reviews
top_25_positive_verb_words, title = top_25_bar(positive_verbs, 'positive',␣
 ↪'verbs')
count += 1
print("FIGURE ", count, "(above): ", title)
```

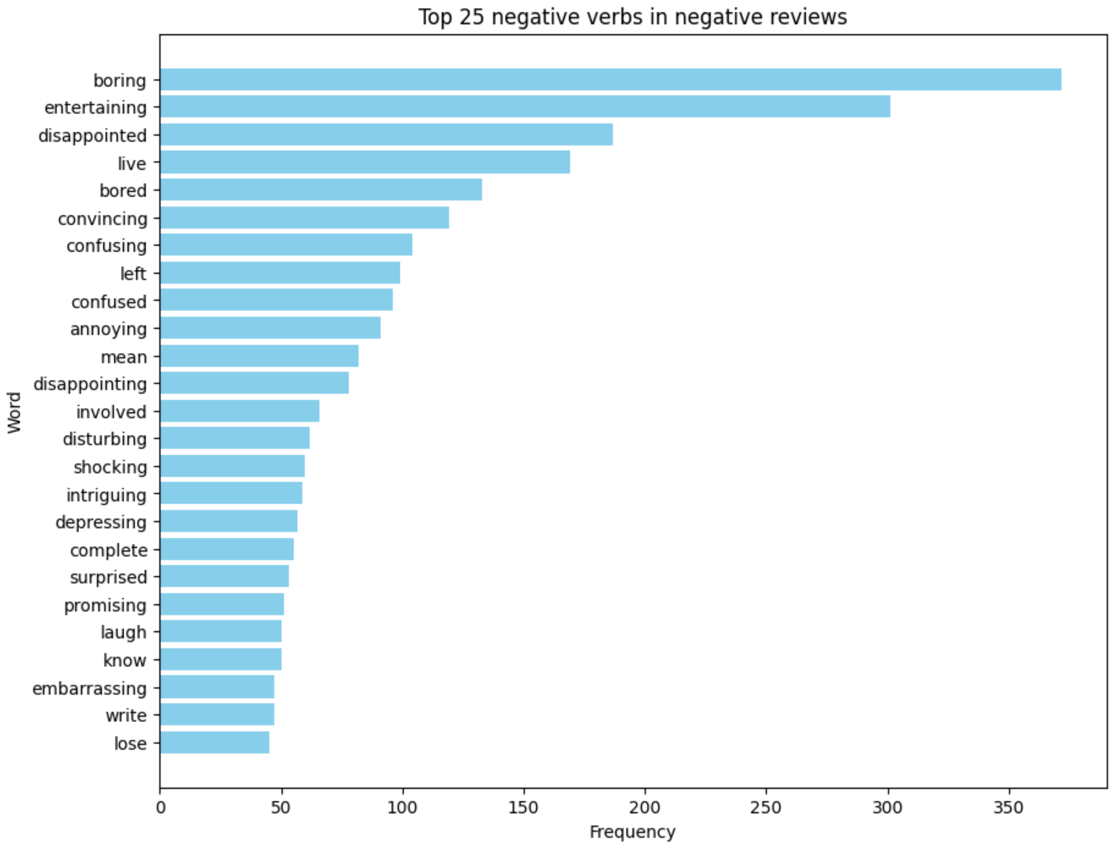


FIGURE 9 (above): Top 25 negative verbs in negative reviews

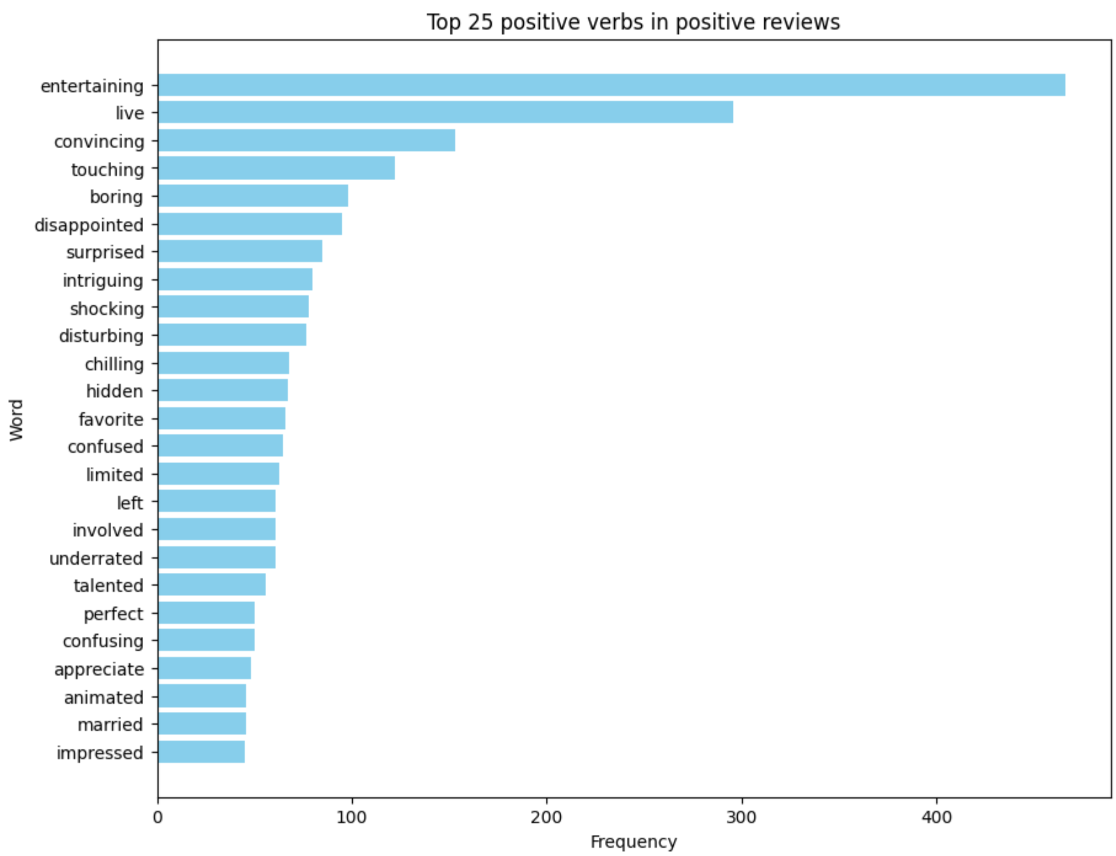


FIGURE 10 (above): Top 25 positive verbs in positive reviews

[29]:
```
###########################################################################################
# DISPLAY WORD CLOUDS SHOWING TOP 25 VERBS FOR POSITIVE AND NEGATIVE REWIEWS␣
↪##############
###########################################################################################

# Top 25 verbs word cloud in negative reviews
title = top_25_wordcloud(negative_verbs, 'NEGATIVE', 'VERBS\n')
count += 1
print("FIGURE ", count, "(above): ", title)
# Top 25 verbs word cloud in positive reviews
title = top_25_wordcloud(positive_verbs, 'POSITIVE', 'VERBS\n')
count += 1
print("FIGURE ", count, "(above): ", title)
```

FIGURE 11 (above): TOP 25 NEGATIVE VERBS
IN NEGATIVE REVIEWS

FIGURE 12 (above): TOP 25 POSITIVE VERBS
IN POSITIVE REVIEWS

Observations: – In negative reviews the verbs ‘boring’, ‘disappointed’, and ‘bored’ ranked high (figures 9 and 11). – In positive reviews the verbs 'convincing' and 'touching' ranked high (figures 10 and 12). – Surprisingly, some verbs appeared in both positive and negative reviews : 'confused/confusing', 'disappointed' and 'boring' (figures 9 and 10). – Based upon the word clouds (figures 11 and 12), it seems both positive and negative reviewers found the movies they watched 'entertaining'. Negative reviewers were more 'bored' and 'disappointed'. Specifically, they may have been 'bored' and 'disappoined' with the 'characters', 'acting' and 'story' as shown in the top 25 general word bar graphs (figure 1).

[30]:
```
###################=============== CLASSIFICATION␣
↪=====================###################

##############################################################################################
################### TRY CLASSIFICATION **WITHOUT** CROSSFOLD VALIDATION␣
↪###################
##############################################################################################

# Split the dataset into texts and labels
texts = df_all_reviews['review'].values
labels = df_all_reviews['sentiment'].values

# Tokenize the text data by converting each word into a vocabulary index number

# Define maximum number of unique words to be considered in the tokenization␣
↪process
#    based upon word frequency (the vocabulary)
max_words = 10000
# Instance of tokenizer class
# oov_token="<OOV>" defines a token to be used for out-of-vocabulary (OOV) words
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
# Acquire the 10000 words in the vocabulary by analyzing each line of text.
# Each word is assigned an integer index in the vocabulary
tokenizer.fit_on_texts(texts)
# Replace each word in the text with its corresponding integer index in the
#   vocabulary. Words that are not present in the vocabulary are replaced with
#   the OOV token specified earlier
sequences = tokenizer.texts_to_sequences(texts)

# Pad sequences to ensure uniform length

# Define maximum number of tokens (or words) that each sequence should contain␣
↪after padding or truncation.
max_length = 100
# Each sequence is a list of integers, with each integer corresponding to a␣
↪word index in the vocabulary.
# Sequences are padded or truncated at their ends or "post" to max length of 100
padded_sequences = pad_sequences(sequences, maxlen=max_length, padding='post',␣
↪truncating='post')
```

```
# Split the data into training and test sets
# Test/Train data sets ratio is 50/50
train_texts, test_texts, train_labels, test_labels =␣
↪train_test_split(padded_sequences, labels, test_size=0.5, random_state=42)
```

[31]:
```
# Define the model architecture
model = tf.keras.Sequential([
    # "input_dim" -- the maximum integer index that can be expected in input␣
↪data
    # "output_dim" -- determines the dimensionality of the embedding space.
    #    Represent each word as a dense vector of length output_dim
    # "input_length" -- specifies the length of input sequences.
    # This dense layer captures word semantics or in text context word meanings
    tf.keras.layers.Embedding(input_dim=max_words, output_dim=16,␣
↪input_length=max_length),
    # This layer averages the feature values across all time steps in each␣
↪sequence
    #   into a shorter sequence for faster processing
    tf.keras.layers.GlobalAveragePooling1D(),
    # The sigmoid activation function compresses outputs of the previous layer␣
↪between
    #   1 and 0, expressed as probabilities for binary classification.
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model with loss function, chosen optimizer, and accuracy metric
model.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])
```

```
2024-04-01 01:37:19.450053: I
tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool
with default inter op setting: 2. Tune using inter_op_parallelism_threads for
best performance.
```

[32]:
```
# Train the model
model.fit(train_texts, train_labels, epochs=10, batch_size=32,␣
↪validation_data=(test_texts, test_labels))
```

```
Epoch 1/10
782/782 [==============================] - 4s 5ms/step - loss: 0.6171 -
accuracy: 0.7632 - val_loss: 0.5102 - val_accuracy: 0.8295
Epoch 2/10
782/782 [==============================] - 3s 4ms/step - loss: 0.4253 -
accuracy: 0.8540 - val_loss: 0.3855 - val_accuracy: 0.8536
Epoch 3/10
782/782 [==============================] - 3s 4ms/step - loss: 0.3341 -
accuracy: 0.8762 - val_loss: 0.3405 - val_accuracy: 0.8637
```

```
Epoch 4/10
782/782 [==============================] - 3s 4ms/step - loss: 0.2891 -
accuracy: 0.8918 - val_loss: 0.3210 - val_accuracy: 0.8691
Epoch 5/10
782/782 [==============================] - 3s 4ms/step - loss: 0.2598 -
accuracy: 0.9013 - val_loss: 0.3124 - val_accuracy: 0.8704
Epoch 6/10
782/782 [==============================] - 4s 5ms/step - loss: 0.2378 -
accuracy: 0.9107 - val_loss: 0.3095 - val_accuracy: 0.8700
Epoch 7/10
782/782 [==============================] - 3s 4ms/step - loss: 0.2197 -
accuracy: 0.9182 - val_loss: 0.3107 - val_accuracy: 0.8704
Epoch 8/10
782/782 [==============================] - 3s 4ms/step - loss: 0.2044 -
accuracy: 0.9249 - val_loss: 0.3139 - val_accuracy: 0.8707
Epoch 9/10
782/782 [==============================] - 3s 4ms/step - loss: 0.1906 -
accuracy: 0.9304 - val_loss: 0.3234 - val_accuracy: 0.8673
Epoch 10/10
782/782 [==============================] - 3s 4ms/step - loss: 0.1785 -
accuracy: 0.9344 - val_loss: 0.3281 - val_accuracy: 0.8678
```

[32]:
```
<keras.callbacks.History at 0x7f96324466b0>
```

[33]:
```python
# Evaluate the model on test set
test_loss, test_accuracy = model.evaluate(test_texts, test_labels)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

```
782/782 [==============================] - 1s 1ms/step - loss: 0.3281 -
accuracy: 0.8678
Test Loss: 0.32805135846138
Test Accuracy: 0.8678399920463562
```

[34]:
```python
###########################################################################################
################### TRY CLASSIFICATION **WITH** CROSSFOLD VALIDATION
 ↪######################
###########################################################################################

# Split the dataset into texts and labels
texts = df_all_reviews['review'].values
labels = df_all_reviews['sentiment'].values

# Tokenize the text data
max_words = 10000
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
```

```python
# Pad sequences to ensure uniform length
max_length = 100
padded_sequences = pad_sequences(sequences, maxlen=max_length, padding='post',␣
↪truncating='post')

# Define the model architecture
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=max_words, output_dim=16,␣
↪input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])

# Define number of folds for cross-validation
# This breaks the train and test data sets into 5 "folds" or sections
num_folds = 5

# Perform cross-validation
# Split train data sets and test data sets into 5 folds each
kf = KFold(n_splits=num_folds, shuffle=True)
# Define an empty array to hold accuracy of each test fold-train fold pair
fold_accuracy = []

for train_index, test_index in kf.split(padded_sequences):
    # Select a train data set fold or section
    train_texts_fold = padded_sequences[train_index]
    train_labels_fold = labels[train_index]
    # Select a test data set fold or section
    test_texts_fold = padded_sequences[test_index]
    test_labels_fold = labels[test_index]

    # Train the model on this test fold-train fold pair
    model.fit(train_texts_fold, train_labels_fold, epochs=10, batch_size=32)

    # Evaluate accuracy on this test fold-train fold pair
    _, accuracy = model.evaluate(test_texts_fold, test_labels_fold)
    fold_accuracy.append(accuracy)

# Compute mean accuracy across all test fold-train fold pairings
mean_accuracy = np.mean(fold_accuracy)
print("Mean Cross-Validation Accuracy:", mean_accuracy)
```

```
Epoch 1/10
1250/1250 [==============================] - 5s 4ms/step - loss: 0.5593 -
accuracy: 0.7895
Epoch 2/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.3545 -
accuracy: 0.8682
Epoch 3/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.2912 -
accuracy: 0.8864
Epoch 4/10
1250/1250 [==============================] - 3s 2ms/step - loss: 0.2598 -
accuracy: 0.8991
Epoch 5/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.2390 -
accuracy: 0.9074
Epoch 6/10
1250/1250 [==============================] - 3s 2ms/step - loss: 0.2231 -
accuracy: 0.9139
Epoch 7/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.2106 -
accuracy: 0.9191
Epoch 8/10
1250/1250 [==============================] - 3s 2ms/step - loss: 0.1997 -
accuracy: 0.9232
Epoch 9/10
1250/1250 [==============================] - 3s 2ms/step - loss: 0.1907 -
accuracy: 0.9275
Epoch 10/10
1250/1250 [==============================] - 3s 2ms/step - loss: 0.1823 -
accuracy: 0.9310
```

[34]: <keras.callbacks.History at 0x7f962cb4e3b0>

```
313/313 [==============================] - 0s 1ms/step - loss: 0.3436 -
accuracy: 0.8664
Epoch 1/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.2163 -
accuracy: 0.9178
Epoch 2/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.2016 -
accuracy: 0.9244
Epoch 3/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1916 -
accuracy: 0.9282
Epoch 4/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1834 -
accuracy: 0.9317
Epoch 5/10
```

```
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1762 -
accuracy: 0.9344
Epoch 6/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1696 -
accuracy: 0.9366
Epoch 7/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1643 -
accuracy: 0.9394
Epoch 8/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1589 -
accuracy: 0.9418
Epoch 9/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1544 -
accuracy: 0.9444
Epoch 10/10
1250/1250 [==============================] - 3s 2ms/step - loss: 0.1499 -
accuracy: 0.9460
```

[34]: <keras.callbacks.History at 0x7f96296506a0>

```
313/313 [==============================] - 0s 884us/step - loss: 0.3073 -
accuracy: 0.8882
Epoch 1/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1878 -
accuracy: 0.9321
Epoch 2/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1741 -
accuracy: 0.9367
Epoch 3/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1656 -
accuracy: 0.9409
Epoch 4/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1595 -
accuracy: 0.9427
Epoch 5/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1543 -
accuracy: 0.9448
Epoch 6/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1496 -
accuracy: 0.9464
Epoch 7/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1454 -
accuracy: 0.9491
Epoch 8/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1414 -
accuracy: 0.9503
Epoch 9/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1383 -
```

```
accuracy: 0.9507
Epoch 10/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1350 -
accuracy: 0.9525
```

[34]: <keras.callbacks.History at 0x7f9626a33640>

```
313/313 [==============================] - 0s 964us/step - loss: 0.2930 -
accuracy: 0.8910
Epoch 1/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1753 -
accuracy: 0.9365
Epoch 2/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1607 -
accuracy: 0.9418
Epoch 3/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1528 -
accuracy: 0.9440
Epoch 4/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1467 -
accuracy: 0.9473
Epoch 5/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1425 -
accuracy: 0.9488
Epoch 6/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1383 -
accuracy: 0.9506
Epoch 7/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1345 -
accuracy: 0.9518
Epoch 8/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1311 -
accuracy: 0.9540
Epoch 9/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1281 -
accuracy: 0.9552
Epoch 10/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1254 -
accuracy: 0.9558
```

[34]: <keras.callbacks.History at 0x7f962a8bb760>

```
313/313 [==============================] - 0s 880us/step - loss: 0.2922 -
accuracy: 0.8926
Epoch 1/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1687 -
accuracy: 0.9405
Epoch 2/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1538 -
```

```
accuracy: 0.9454
Epoch 3/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1464 -
accuracy: 0.9480
Epoch 4/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1408 -
accuracy: 0.9509
Epoch 5/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1365 -
accuracy: 0.9518
Epoch 6/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1328 -
accuracy: 0.9538
Epoch 7/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1296 -
accuracy: 0.9539
Epoch 8/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1267 -
accuracy: 0.9561
Epoch 9/10
1250/1250 [==============================] - 4s 3ms/step - loss: 0.1241 -
accuracy: 0.9567
Epoch 10/10
1250/1250 [==============================] - 3s 3ms/step - loss: 0.1213 -
accuracy: 0.9584
```

[34]:
```
<keras.callbacks.History at 0x7f962ca7f070>
```

```
313/313 [==============================] - 0s 908us/step - loss: 0.2781 -
accuracy: 0.8980
Mean Cross-Validation Accuracy: 0.8872399926185608
```

[35]:
```
# Evaluate the cross validation trained model on test set
test_loss, test_accuracy = model.evaluate(test_texts, test_labels)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

```
782/782 [==============================] - 1s 954us/step - loss: 0.1364 -
accuracy: 0.9552
Test Loss: 0.13642701506614685
Test Accuracy: 0.9551600217819214
```

[36]:
```
# Create confusion matrix
predicted_labels = model.predict(test_texts)
binary_predictions = (predicted_labels > 0.5).astype(int)
# true labels are test labels
# final predicted labels are binary predictions
cm = confusion_matrix(test_labels, binary_predictions)
```

```
782/782 [==============================] - 1s 705us/step
```

[37]:
```python
print(test_labels)
print(len(test_labels))
```

```
[1 1 0 … 1 0 0]
25000
```

[38]:
```python
print(binary_predictions)
print(len(binary_predictions))
```

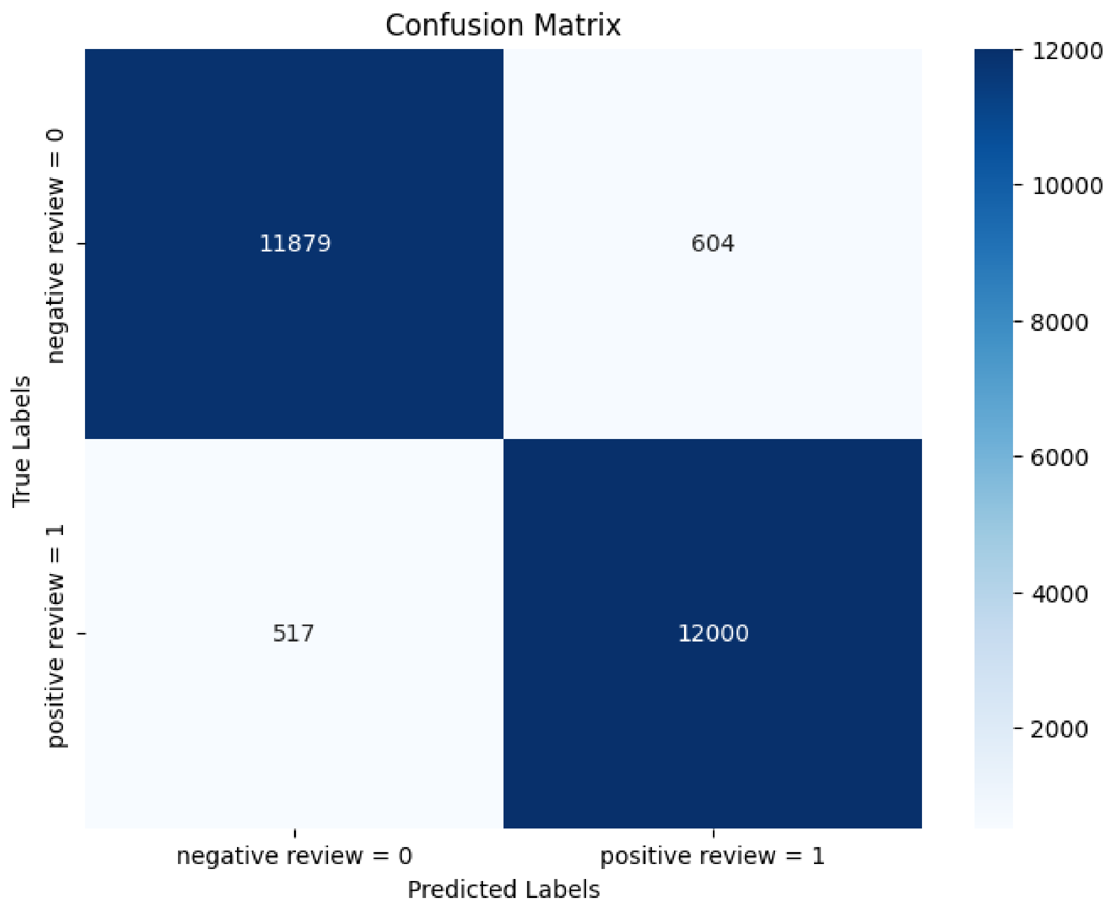
```
[[1]
 [1]
 [0]
 …
 [1]
 [0]
 [0]]
25000
```

[39]:
```python
# Flatten the binary_predictions and convert it to a list
flattened_binary_predictions = binary_predictions.flatten().tolist()
```

[40]:
```python
###########################################################################################
# PLOT CONFUSION MATRIX␣
 ↪###################################################################
###########################################################################################

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
classes = ['negative review = 0','positive review = 1']
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,␣
 ↪yticklabels=classes)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show();

misclassified_indices = np.where(test_labels !=␣
 ↪flattened_binary_predictions)[0];
print("Number of mislabeled reviews:", len(misclassified_indices));
print("Number of predictions:", len(predicted_labels));
print("Overall accuracy:", round((1- (len(misclassified_indices)/
 ↪len(predicted_labels)))*100, 2), "percent" );
```




```
Number of mislabeled reviews: 1121
Number of predictions: 25000
Overall accuracy: 95.52 percent
```

[43]:

```
###########################################################################################
# SHOW SAMPLE OF MISCLASSIFIED REVIEWS␣
↪#####################################################
###########################################################################################

# Display some of the misclassified sentiment reviews
misclassified_indices = np.where(test_labels != flattened_binary_predictions)[0]
# Sample size limited to 10 reviews
num_misclassified_reviews = min(10, len(misclassified_indices))
print("Number of mislabeled :", len(misclassified_indices),"\n");
print("Misclassified Reviews:\n")
for i in range(num_misclassified_reviews):
    idx = misclassified_indices[i]
    # print("Review:", texts[idx])
```

```
    print("Review:", df_original_reviews['review'].values[idx]) # show original␣
↪pre processed texts
    print("True Label:", test_labels[idx])
    print("Predicted Label:", binary_predictions[idx])
    print()
```

```
Number of mislabeled : 1121

Misclassified Reviews:

Review: film laboured along predictable story lines shallow characters ever seen
writer obviously bought playbook write space disaster movie followed play play
particular stereotypical use astronauts talking loved ones outer space putting
brave show face disaster done time time againMax Q appears written hope
producers would throw $ 50 million project judging latter half film contained
numerous lame attempts special effects producers could muster $ 50 thousand
learn film nominated Special Visual Effects Emmy absolutely gobsmackedI think
handful high school students pass Media Studies could created believable
effectsAnd plot holes numerous mention pick one example Im NASA expert surely
highly implausible worker attached shuttle simulator would suddenly hold
position power control room things start go pearshaped program Surely someone
experienced Mission Control Program Director would call rather twentynine year
old control room beforeThe saving grace film work Bill Campbell manages make
good attempt salvaging something train wreck scriptI give film 2 10 aboveaverage
work Bill Campbell lead role saving lower mark
True Label: 0
Predicted Label: [1]

Review: find intriguing Lee Radziwill Jackie Kennedy sister cousin women would
encourage Maysles make Big Edie Little Edie subject film certainly could
considered skeletons family closet extra features DVD include several
contemporary fashion designers crediting ideas oddball women Id say anyone
interested fashion would find discussion designers fascinating ie nuts missing
something movie hard come Netflix Facets though
True Label: 0
Predicted Label: [1]

Review: Ye Lou film Purple Butterfly pits secret organization Purple Butterfly
Japanese forces war torn Shanghai Ding Hui Zhang Ziyi exlover Hidehiko Itami
Toru Nakamura find opposite sides conflict chance meetingI agree reviewer Paris
film substitutes convoluted semihistorical conflict plot without giving audience
single reason care characters causes sudden time shifting help matters appears
completely unwarranted pointless Normally mind dark movies absence light
bonejarringly shaky camera footage generally bad filmmaking techniques really
make tough film watch stay interested also agree viewer Georgia film chaotic
editing style claustrophobic cinematography think helps movie backdrop film one
potent events 20th Century believe justice editing Michael Bay film overly
melodramatic moments add watchabilityThe actors suitably melancholy Zhang Ziyi
```

shows exceptionally limited acting range spends entire movie seems best films brooding looking generally annoyed However least adds variety role chainsmoking engaging worst lovemaking scene since Michael Biehn Linda Hamilton TerminatorAll disappointing film especially seeing comes director Suzhou 2/10
True Label: 1
Predicted Label: [0]

Review: hearing George Orwell prophetic masterpiece life Im 37 never read book totally confused Ive seenI familiar concepts covered novel im sure hearsay quotes Without limited knowledge film would complete mystery even Im still educated story 1984 watched itOn plus sideThe cinematography amazing Hurt & Burton deliver fine performances overall feel movie wonderfully grim desolate prostitute scene fantastically dark piece film makingNow sides plentyThere war going least far propaganda concerned & Nothing explained couple names bandied Eurasia etc mean nothing without explanationWho Winston come work changing news reports front line eat food canteen drink drinking entire film weak & ill brainwashed like rest deal mother & sister happened father little back story would nice scrub essential like read book Without confusing hard follow arthouse movie constantly keeps guessing actually going onThe soundtrack disjointed badly edited constant chatter Big Brother screens swamps dialogue places making even harder work whats going accept may artistic choice annoying sameAlso know mentioned nudity seemed totally gratuitous felt like thrown make lack plot coverageI personally ca abide way Hollywood feels explain story lines word word days brainwashed simpletons steps far way imagine totally relies fact youve read book film really literal translation Ive seen many people say would find hard understand 1984 hailed classic isThere denying light years ahead time pretty much predicted every change society date maybe sort bible powers many scifi novelists done without leaving gaping holes storylineI guess done start buy copy book im make sense thisAll disappointed something Ive waited years watch
True Label: 1
Predicted Label: [0]

Review: movie well directed almost totally disregarded bookI guess trying 2 save time upside 2 actor played finny cute dialog main characters appeared little gay case book Major parts book chopped outYou lost effect haunting book left lacking severely Also strong language although brief unnecessary Also surprised pleasantly new character bookOne favorite characters leper poorly interpreted portrayed seemed sinister movie real leper book disappointing
True Label: 0
Predicted Label: [1]

Review: movie took surprise opening credit sequence features nicely done animation plunged semicheesy production betraying low budget characters typical American teens introduced slowly personal detail usually found movies like time shlitz hits fan know one characters either like hate according distinct personalities slow uphill setup kind like ride slope really tall roller coaster Thankfully action kicks full blown old school HORROR Steve Johnson makeup effects awesome Equal quality much bigger budgeted films scares jolting Kevin

Tenney delivers best movie ever heartstopping surprises creepy suspenseful setups tongueincheek sometimes cheesy humor marks film pure 80s horror opposed sullen tone earlier genre fare like Night Living Dead Hills Eyes true horror fans one worth checking Play first entry double bill 1999 remake House Haunted Hill setup character dynamics similar really wonder film actually remaking
True Label: 1
Predicted Label: [0]

Review: keep rigid historical perspective film actually quite entertaining got action adventure romance one premiere casting matchups era Errol Flynn Olivia de Havilland lead roles evident board picture pass muster purists look one hundred percent accuracy story telling get beyond one need put aside history book enjoy story work fiction know know hard consider Custer Last Stand Little Big Horn prominence history post Civil War America guess unresolved quandary picture matter look itThere lot take though picture two hour plus run time Custer arrival West Point probably first head scratcher riding full military regalia practical joke Sharp Arthur Kennedy putting Major headquarters probably gotten troubleIronically lot scenes military film play comedy Custer first meeting Libby Bacon subsequent encounters include tea reader Callie Hattie McDaniel noticed films McDaniel reminded awful lot another favorite character actor mine Forties Mantan Moreland much one scene looked like might Moreland hamming dress mind owl scene hoot tooAs Flynn interesting note year earlier portrayed JEB Stuart opposite Ronald Reagan depiction General Custer Santa Fe Trail vying attention none Olivia de Havilland film Reagan put none arrogance flamboyance character Custer history remembers Flynn portrayal evident come close Richard Mulligan take military hero 1970 Little Big Man Let say one bit topThe better take away picture manner Custer persevered maintain good name gamble away risky business venture loyalty men led battle along discipline developed course story poignant final confrontation arch rival Sharp riding Little Big Horn declared hell glory entirely dependent one point view Earlier similar remark might given us best insight Custer character stated take glory time go
True Label: 1
Predicted Label: [0]

Review: film quickly gets major chase scene ever increasing destruction first really bad thing guy hijacking Steven Seagal would beaten pulp Seagal driving probably would ended whole premise movieIt seems like decided make kinds changes movie plot plan enjoy action expect coherent plot Turn sense logic may reduce chance getting headacheI give hope Steven Seagal trying move back towards type characters portrayed popular movies
True Label: 1
Predicted Label: [0]

Review: Based Edgar Rice Burroughs novel EARTHS CORE provides little means escape give brain rest Victorian scientist Dr Abner PerryPeter Cushinginvents giant burrowing machine American partnerDoug McClureuse corkscrew way deep earth explore mysteries may hold soon discover lost world subhuman creatures conflict prehistoric monstersCushing comes across absent minded professor point annoying

```
Instead bold adventurer comes across effeminate hand McClure overacted enough
make also laughable Caroline Munro plays pretty Princess Dia refuses leave world
near center earth Also cast Godfrey James Cy Grant Michael Crane
True Label: 0
Predicted Label: [1]

Review: read review Alex Sander sic rather looking rating 6 select choice
ignorant viewing public would seen desecration Alien fantastic dramatic well
made horror/scifi Predator great scifi/action messabout really blame though saw
Alien versus Predator average grading 6 stars connoisseurs film frequent
siteSTOP READING FEAR EVER SUSPENSE RIDDEN PLOT RUINED YOURight beginning film
ridiculous explanation offered Predator ship overrun/not overrun Aliens OK maybe
going throw aliens Earth hunt something went wrong result Alien/Predator hybrid
rest crew realise sooner despite great technology start actually coherent
interesting part film idea perhaps gets really ridiculous always leave disbelief
strictly suspended door screen entering collect way could hereA father son
hunting woods damaged ship crash lands view given would calculate least 10 odd
miles away thick woodland man boy track alone find ship get face hugged Even
point feel little mainly face huggers almost comical rather scary movement
actions father seems like irresponsible dumb redneck muppetAn edgy thrillertype
scenario introduced excon returning town near crash site met somewhat
emotionless dull cop friend bus say introduced mean feeble attempt crap actors
feeling played slasher/horror element introduced sexy girl usual supposedly
nerdy somehow undesirable cute guy gets beaten protective crazy nasty Jock type
American sportsman Scottish man Oh cute/not cute boy excon brother way Yes
theyre clever director brothers whose name research order avoid shite put modern
role reversal oh boring attempt PC Ripley credential type character introduction
comes female soldier returning home husband childGuess happens next wo tell much
actual smiles sadly demise storytelling large majority recent films plot case
got far brightest star Alienridden universeThe Predator stupid reasons stated
previous poster whose post read late Aliens boring PredatorAlien ridiculous
action times exploitative gratuitous disgusting nonsense hospital scene pregnant
mothers Oh shocked alright Shocked low people go get scare shock titillate
perverse really wanted shock titillate scare people pregnant expecting fathers
souls Alien/Predator shagging saucy women teenage girls rather killing
characters depth neither plot filmed paced badly acted disinterested people
blame tarnishes two rather interesting good sets scifi characters film rubbish
gain enjoyment really worry seen well please make decisionPS even mention way
trained soldiers killed 20 seconds amateur civilians survive throughout
True Label: 1
Predicted Label: [0]
```

[ ]:

```

```