```
!pip install pyspellchecker
```

```
Collecting pyspellchecker
  Downloading pyspellchecker-0.8.2-py3-none-any.whl.metadata (9.4 kB)
Downloading pyspellchecker-0.8.2-py3-none-any.whl (7.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.1/7.1 MB 119.3 MB/s eta 0:00:00
Installing collected packages: pyspellchecker
Successfully installed pyspellchecker-0.8.2
```

```
!pip install contractions
```

```
Collecting contractions
  Downloading contractions-0.1.73-py2.py3-none-any.whl.metadata (1.2 kB)
Collecting textsearch>=0.0.21 (from contractions)
  Downloading textsearch-0.0.24-py2.py3-none-any.whl.metadata (1.2 kB)
Collecting anyascii (from textsearch>=0.0.21->contractions)
  Downloading anyascii-0.3.2-py3-none-any.whl.metadata (1.5 kB)
Collecting pyahocorasick (from textsearch>=0.0.21->contractions)
  Downloading pyahocorasick-2.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Downloading contractions-0.1.73-py2.py3-none-any.whl (8.7 kB)
Downloading textsearch-0.0.24-py2.py3-none-any.whl (7.6 kB)
Downloading anyascii-0.3.2-py3-none-any.whl (289 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 289.9/289.9 kB 22.4 MB/s eta 0:00:00
Downloading pyahocorasick-2.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (118 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.3/118.3 kB 8.5 MB/s eta 0:00:00
Installing collected packages: pyahocorasick, anyascii, textsearch, contractions
Successfully installed anyascii-0.3.2 contractions-0.1.73 pyahocorasick-2.1.0 textsearch-0.0.24
```

```
 import pandas as pd
 import numpy as np
 import matplotlib.pyplot as plt
 import seaborn as sns
 import re
 import nltk
from nltk.corpus import stopwords
 from nltk.tokenize import word_tokenize
 from nltk.stem import WordNetLemmatizer
 from contractions import fix
```

```
df = pd.read_csv('/content/Train.csv')
df.head()
```

```
                                          text  label
0    I grew up (b. 1965) watching and loving the Th...      0
1    When I put this movie in my DVD player, and sa...      0
2  Why do people who do not know what a particula...      0
3        Even though I have great interest in Biblical ...      0
4      Im a die hard Dads Army fan and nothing will e...      1
```

```
df.info()
```

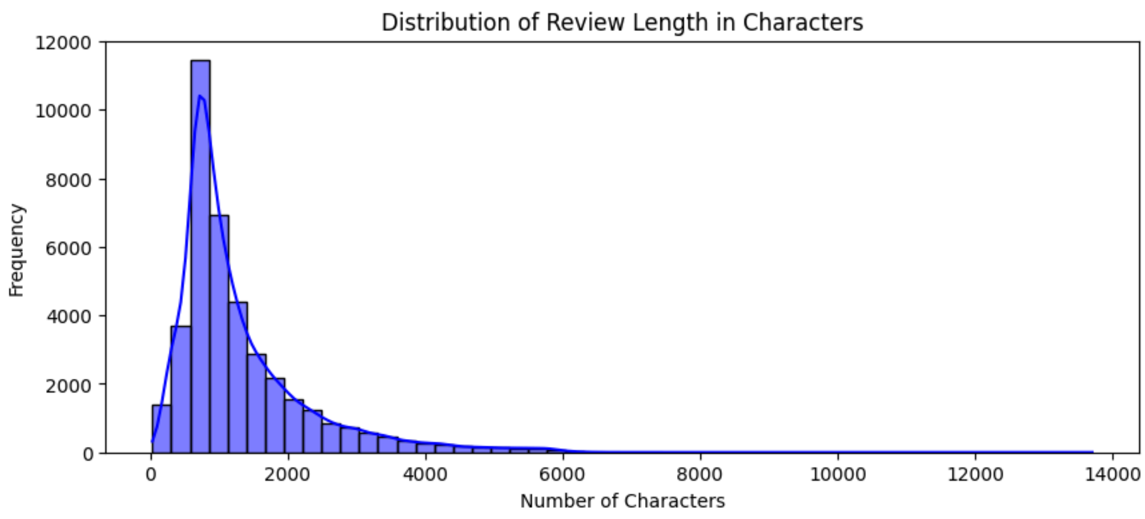
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    40000 non-null  object
 1   label   40000 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 625.1+ KB
```

```
df['text'][1]
```

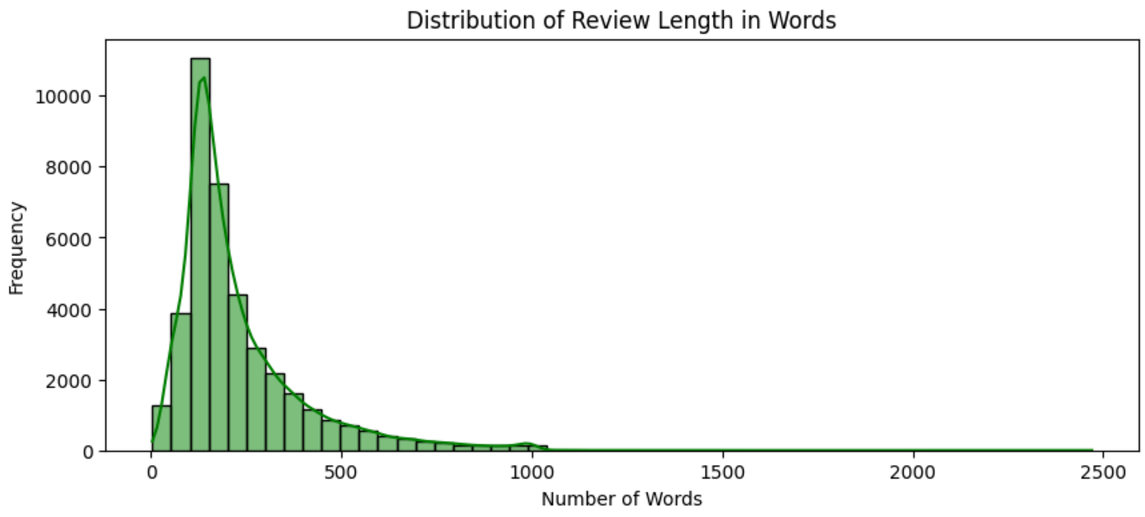
```
'When I put this movie in my DVD player, and sat down with a coke and some chips, I had some expectations. I was hoping that this movie
would contain some of the strong-points of the first movie: Awsome animation, good flowing story, excellent voice cast, funny comedy an
d a kick-ass soundtrack. But, to my disappointment, not any of this is to be found in Atlantis: Milo's Return. Had I read some reviews
first, I might not have been so let down. The following paragraph will be directed to those who have seen the first movie, and who enjo
yed it primarily for the points mentioned.<br /><br />When the first scene appears, your in for a shock if you just picked Atlantis: Mi
lo's Return from the display-case at your local videoshop (or whatever), and had the expectations I had. The music feels as a bad imita
```

```
plt.figure(figsize=(10,4))
df['review_length_char'] = df['text'].apply(len)
```

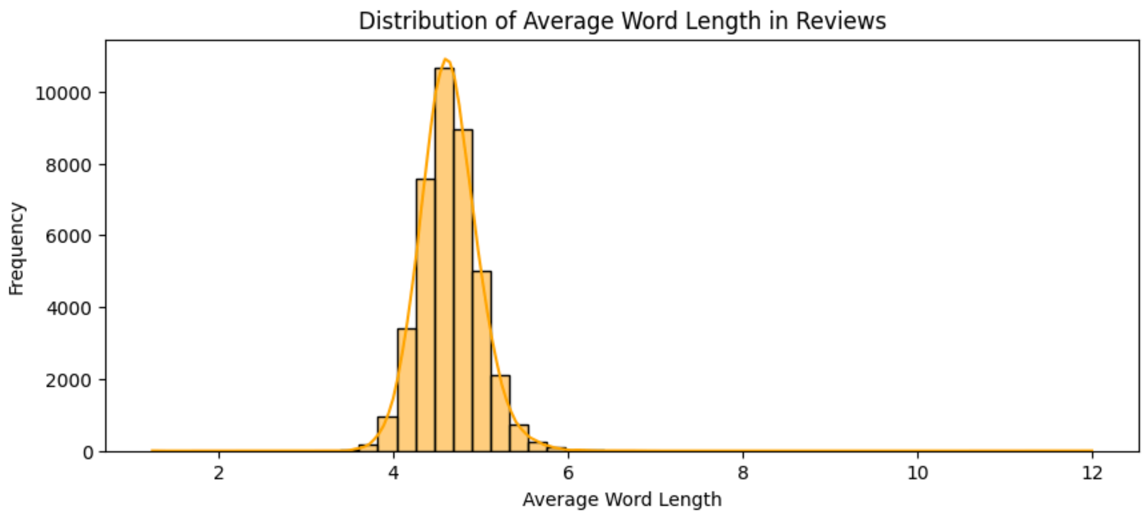
```
sns.histplot(df['review_length_char'], bins=50, kde=True, color='blue')
plt.title("Distribution of Review Length in Characters")
plt.xlabel("Number of Characters")
plt.ylabel("Frequency")
plt.show()
```



```
plt.figure(figsize=(10,4))
df['review_length_word'] = df['text'].apply(lambda x: len(x.split()))
sns.histplot(df['review_length_word'], bins=50, kde=True, color='green')
plt.title("Distribution of Review Length in Words")
plt.xlabel("Number of Words")
plt.ylabel("Frequency")
plt.show()
```



```
df['avg_word_length'] = df['text'].apply(lambda x: np.mean([len(word) for word in x.split()]))
plt.figure(figsize=(10,4))
sns.histplot(df['avg_word_length'], bins=50, kde=True, color='orange')
plt.title("Distribution of Average Word Length in Reviews")
plt.xlabel("Average Word Length")
plt.ylabel("Frequency")
plt.show()
```

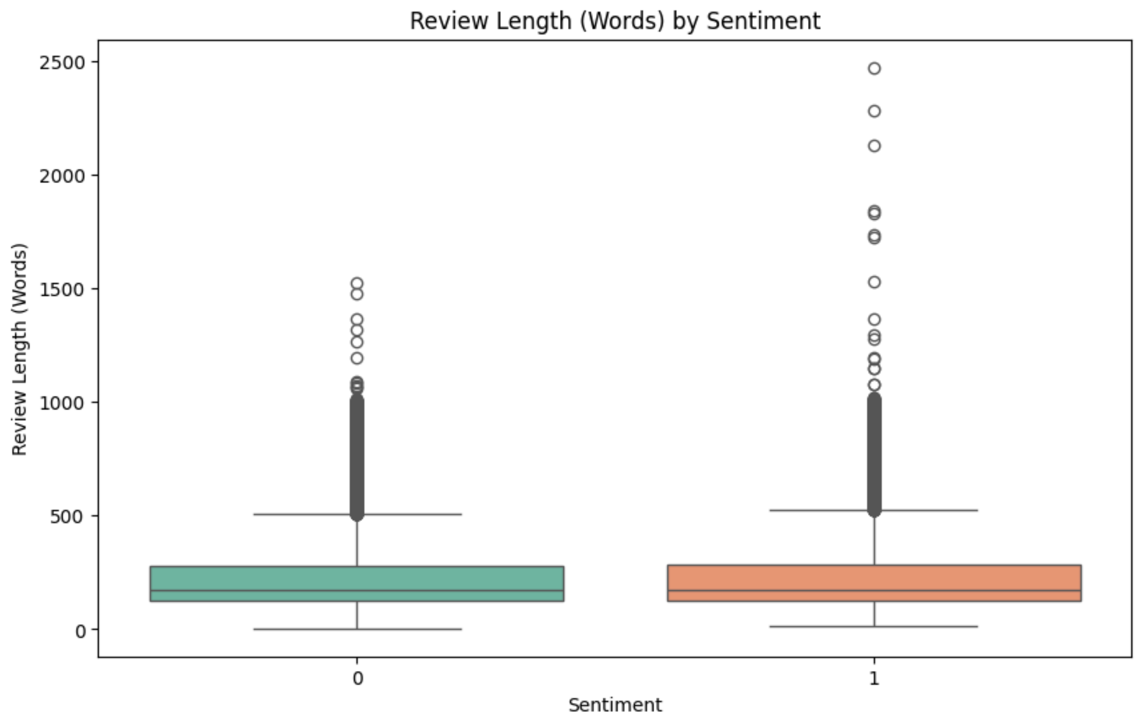


```
print(df['label'].unique())
positive_reviews = df[df['label'] == 'pos']['text']
negative_reviews = df[df['label'] == 'neg']['text']
```

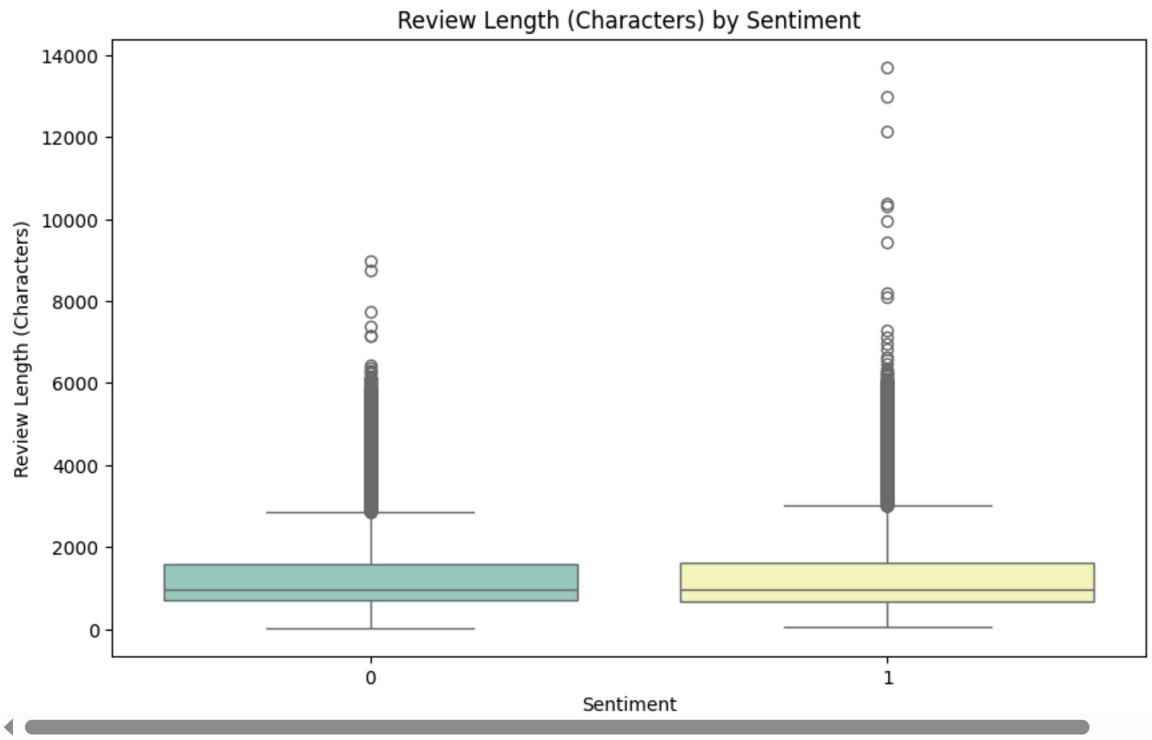
```
[0 1]
```

```
plt.figure(figsize=(10,6))

sns.boxplot(x='label', y='review_length_word', data=df, palette="Set2")
plt.title("Review Length (Words) by Sentiment")
plt.xlabel("Sentiment")
plt.ylabel("Review Length (Words)")
plt.show()

plt.figure(figsize=(10,6))
# Changed 'sentiment' to 'label' to use the existing column for sentiment
sns.boxplot(x='label', y='review_length_char', data=df, palette="Set3")
plt.title("Review Length (Characters) by Sentiment")
plt.xlabel("Sentiment")
plt.ylabel("Review Length (Characters)")
plt.show()
```

```
<ipython-input-13-5c4a4632a18a>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x='label', y='review_length_word', data=df, palette="Set2")
```



```
<ipython-input-13-5c4a4632a18a>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `leg

  sns.boxplot(x='label', y='review_length_char', data=df, palette="Set3")
```



```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
```

```
from spellchecker import SpellChecker
from contractions import fix

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
   # Lowercase the text
   text = text.lower()
   # Expand contractions
   text = fix(text)

   text = re.sub(r'<.*?>', '', text)
   text = re.sub(r'https?://\S+|www\.\S+', '', text)
   # Remove special characters (keeping words, spaces, hyphens, apostrophes)
   text = re.sub(r"[^\w\s'-]", ' ', text)
   # Tokenize the text
   tokens = word_tokenize(text)
   # Remove stopwords and lemmatize
   tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
   # Join tokens back into a single string
   return ' '.join(tokens)
df['text'] = df['text'].apply(preprocess_text)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
<ipython-input-14-35c036a5aa6d> in <cell line: 0>()
     30    # Join tokens back into a single string
     31    return ' '.join(tokens)
---> 32 df['text'] = df['text'].apply(preprocess_text)

 11 frames
lib.pyx in pandas._libs.lib.map_infer()

/usr/local/lib/python3.11/dist-packages/nltk/data.py in find(resource_name, paths)
    577     sep = "*" * 70
    578     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579     raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource punkt_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt_tab/english/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

```
df['text'][1]
```

```
df['text'].groupby(df['label']).count()
```

	text
label	
0	20019
1	19981

dtype: int64

```
df.groupby('label').describe()
```

	review_length_char								review_length_word					avg_word_length		
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std
label																
0	20019.0	1292.536990	942.220087	32.0	705.0	973.0	1571.0	8969.0	20019.0	229.204606	...	279.0	1522.0	20019.0	4.623670	0.328
1	19981.0	1328.083279	1032.236721	65.0	690.0	972.0	1621.0	13704.0	19981.0	233.477954	...	285.0	2470.0	19981.0	4.657509	0.350

2 rows × 24 columns

```
df['length'] = df['text'].apply(len)
df.head()
```

	text	label	review_length_char	review_length_word	avg_word_length	length
0	I grew up (b. 1965) watching and loving the Th...	0	874	151	4.794702	874
1	When I put this movie in my DVD player, and sa...	0	1811	326	4.558282	1811
2	Why do people who do not know what a particula...	0	983	184	4.347826	983
3	Even though I have great interest in Biblical ...	0	351	69	4.101449	351
4	Im a die hard Dads Army fan and nothing will e...	1	983	178	4.528090	983

Vectorization

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
bow = CountVectorizer(analyzer='word').fit(df['text'])
```

```
print(len(bow.vocabulary_))
```

```
92908
```

```
MESS2 = df['text'][1]
print(MESS2)
```

```
When I put this movie in my DVD player, and sat down with a coke and some chips, I had some expectations. I was hoping that this movie w
```

```
MESS2 = bow.transform([MESS2])
print(MESS2)
```

```
<Compressed Sparse Row sparse matrix of dtype 'int64'
	with 167 stored elements and shape (1, 92908)>
  Coords	Values
  (0, 2100)	1
  (0, 2438)	1
  (0, 3495)	1
  (0, 3545)	1
  (0, 3593)	1
  (0, 4083)	7
  (0, 4287)	2
  (0, 4644)	1
  (0, 4846)	1
  (0, 5580)	3
  (0, 5733)	1
  (0, 5928)	1
  (0, 6001)	2
  (0, 6181)	1
  (0, 6578)	1
  (0, 6868)	2
  (0, 7938)	3
  (0, 8207)	4
  (0, 8897)	1
  (0, 10945)	4
  (0, 11194)	1
  (0, 12410)	4
  (0, 12510)	1
  (0, 13371)	1
  (0, 13633)	1
  :	:
  (0, 82218)	1
  (0, 82245)	2
  (0, 82262)	20
  (0, 82529)	6
  (0, 82591)	1
  (0, 82642)	2
  (0, 82967)	2
  (0, 83225)	5
  (0, 85169)	1
  (0, 85227)	1
  (0, 88210)	1
  (0, 88378)	1
  (0, 88851)	3
  (0, 89510)	1
  (0, 89717)	1
  (0, 89718)	1
  (0, 90160)	1
  (0, 90214)	2
  (0, 90401)	3
  (0, 90645)	1
  (0, 90964)	2
  (0, 91425)	1
  (0, 91428)	1
  (0, 92197)	6
  (0, 92222)	2
```

```
MESS2.shape
```

```
(1, 92908)
```

```
 bow.get_feature_names_out()[78]
```

```
'1050'
```

```
mess_bow = bow.transform(df['text'])
```

```
print('shape of sparse matrix : ',mess_bow.shape)
print('ammount of non_zero ocarences: ',mess_bow.nnz)
```

```
shape of sparse matrix :  (40000, 92908)
ammount of non_zero ocarences:  5463770
```

```
sparsity = (100.0*mess_bow.nnz/(mess_bow.shape[0]*mess_bow.shape[1]))
print('sparsity : {}'.format(sparsity))
```

```
sparsity : 0.14702097774142162
```

```
 from sklearn.feature_extraction.text import TfidfTransformer
```

```
TF = TfidfTransformer().fit(mess_bow)
```

```
TF.idf_[bow.vocabulary_['bad']]
```

```
np.float64(2.4440544114586342)
```

```
df['label'] = df['label'].map({'pos': 1, 'neg': 0})
```

```
df['label']
```

```
       label
    0    NaN
    1    NaN
    2    NaN
    3    NaN
    4    NaN
  ...    ...
39995    NaN
39996    NaN
39997    NaN
39998    NaN
39999    NaN
40000 rows × 1 columns

dtype: float64
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['label'])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer # Import CountVectorizer here as well

count_vectorizer = CountVectorizer(max_features=5000)

tfidf_vectorizer = TfidfVectorizer(max_features=5000)
```

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
# Download the missing punkt_tab resource
nltk.download('punkt_tab')

from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re

from spellchecker import SpellChecker
from contractions import fix

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
   # Lowercase the text
   text = text.lower()
   # Expand contractions
   text = fix(text)

   text = re.sub(r'<.*?>', '', text)
   text = re.sub(r'https?://\S+|www\.\S+', '', text)
   # Remove special characters (keeping words, spaces, hyphens, apostrophes)
   text = re.sub(r"[^\w\s'-]", ' ', text)
   # Tokenize the text
   tokens = word_tokenize(text)
   # Remove stopwords and lemmatize
   tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
   # Join tokens back into a single string
   return ' '.join(tokens)
```

```
X_train_tokens = [word_tokenize(text) for text in X_train]
X_test_tokens = [word_tokenize(text) for text in X_test]
```

```
# You also need to import Word2Vec
from gensim.models import Word2Vec


word2vec_model = Word2Vec(sentences=X_train_tokens, vector_size=100, window=5, min_count=1, workers=4)

# Function to average word vectors for a document
def document_vector(doc, model):
    # remove out-of-vocabulary words
    words = [word for word in doc if word in model.wv.index_to_key]
    if not words:
        return np.zeros(model.vector_size)
    return np.mean(model.wv[words], axis=0)

# Create document vectors for training and testing data
X_train_w2v = np.array([document_vector(doc, word2vec_model) for doc in X_train_tokens])
X_test_w2v = np.array([document_vector(doc, word2vec_model) for doc in X_test_tokens])

# Now X_train_w2v and X_test_w2v are defined and can be used for training the model

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Train Model
clf = LogisticRegression(max_iter=500)
# Use the newly created Word2Vec features for training
clf.fit(X_train_w2v, y_train)

# Predict on Test Set
# Use the newly created Word2Vec features for prediction
y_pred = clf.predict(X_test_w2v)

# Calculate Accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy with Word2Vec features: {accuracy}")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-58-4a55ae091498> in <cell line: 0>()
     41 
     42 # You also need to import Word2Vec
---> 43 from gensim.models import Word2Vec
     44 
     45 

                                    5 frames
/usr/local/lib/python3.11/dist-packages/gensim/_matutils.pyx in init gensim._matutils()

ValueError: numpy.dtype size changed, may indicate binary incompatibility. Expected 96 from C header, got 88 from PyObject
```

```
# Fit and Transform
X_train_counts = count_vectorizer.fit_transform(X_train)
X_test_counts = count_vectorizer.transform(X_test)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Implementing LSTM (Long Short-Term Memory) for Sentiment Analysis on Textual Data

```
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['label'])
```

```
y_train
```

	label
38125	NaN
2247	NaN
26908	NaN
29934	NaN
34861	NaN
...	...
38804	NaN
9727	NaN
8518	NaN
32495	NaN
1093	NaN

30000 rows × 1 columns

dtype: float64

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional

# Ensure necessary libraries are installed and updated
!pip install tensorflow
!pip install opencv-python
!pip install numpy --upgrade --force-reinstall

# Tokenization Parameters
max_vocab_size = 20000  # Maximum number of words to keep
max_sequence_length = 300  # Maximum length of each review

# Tokenize Text
tokenizer = Tokenizer(num_words=max_vocab_size, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.13.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37.1
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.9)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.15.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.4
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (202
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (3.
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (3.
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow) (3
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->te
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-packages (from opencv-python) (1.26.4)
Collecting numpy
  Using cached numpy-2.2.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (62 kB)
Using cached numpy-2.2.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.4 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
gensim 4.3.3 requires numpy<2.0,>=1.18.5, but you have numpy 2.2.5 which is incompatible.
tsfresh 0.21.0 requires scipy>=1.14.0; python_version >= "3.10", but you have scipy 1.13.1 which is incompatible.
numba 0.60.0 requires numpy<2.1,>=1.22, but you have numpy 2.2.5 which is incompatible.
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 2.2.5 which is incompatible.
Successfully installed numpy-2.2.5
```

```
 X_train_seq = tokenizer.texts_to_sequences(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
# Pad Sequences (Ensure all inputs are of the same length)
X_train_padded = pad_sequences(X_train_seq, maxlen=max_sequence_length, padding='post')
X_test_padded = pad_sequences(X_test_seq, maxlen=max_sequence_length, padding='post')
```

```
X_test_padded
```

```
array([[ 533, 1469,   23, ...,    0,    0,    0],
       [   2,  877,  484, ...,    0,    0,    0],
       [  11,  238,  494, ...,    0,    0,    0],
       ...,
       [  12,   16,    7, ...,    0,    0,    0],
       [1753,  198,   30, ...,    0,    0,    0],
       [  70,   11,  457, ...,    0,    0,    0]], dtype=int32)
```

Define LSTM model

```
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional, GRU
```

```
model = Sequential()
model.add(Embedding(input_dim=max_vocab_size, output_dim=128, input_length=max_sequence_length))
model.add(LSTM(128, dropout=0.2, recurrent_dropout = 0.2, return_sequences=True))
model.add(Bidirectional(GRU(64, return_sequences=False)))
model.add(Dense(1, activation = "sigmoid"))
```