DSC_550_Week_3

March 31, 2024

[3]:
```
# Installs
import pandas as pd
import zipfile
from textblob import TextBlob
from sklearn.metrics import accuracy_score
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

[5]:
```
# Load the movie review data
df = pd.read_csv('/Users/nickblackford/Desktop/Python/labeledTrainData.tsv',␣
 ↪delimiter='        ')
df.head()
```

[5]:

	id	sentiment	review
0	5814_8	1	With all this stuff going down at the moment w…
1	2381_9	1	\The Classic War of the Worlds\" by Timothy Hi…
2	7759_3	0	The film starts with a manager (Nicholas Bell)…
3	3630_4	0	It must be assumed that those who praised this…
4	9495_8	1	Superbly trashy and wondrously unpretentious 8…

[6]:
```
# Counting the number of positive and negative reviews
positive_reviews = df[df['sentiment'] == 1].shape[0]
negative_reviews = df[df['sentiment'] == 0].shape[0]
print(f"Number of positive reviews: {positive_reviews}")
print(f"Number of negative reviews: {negative_reviews}")
```

```
Number of positive reviews: 12500
Number of negative reviews: 12500
```

[7]:
```
# Using TextBlob to classify each movie review
df['predicted_sentiment'] = df['review'].apply(lambda x: 1 if TextBlob(x).
 ↪sentiment.polarity >= 0 else 0)

# Checking the accuracy of the TextBlob model
accuracy_textblob = accuracy_score(df['sentiment'], df['predicted_sentiment'])
print(f"Accuracy of TextBlob model: {accuracy_textblob}")
```

```
Accuracy of TextBlob model: 0.68524
```

[12]:
```python
# EXTRA CREDIT Using VADER for sentiment analysis
from nltk.sentiment import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
df['vader_sentiment'] = df['review'].apply(lambda x: 1 if sia.
 ↪polarity_scores(x)['compound'] >= 0 else 0)
accuracy_vader = accuracy_score(df['sentiment'], df['vader_sentiment'])
print(f"Accuracy of VADER model: {accuracy_vader}")
```

```
Accuracy of VADER model: 0.69356
```

[15]:
```python
# Part 2: Prepping Text for a Custom Model
import re
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

# Convert all text to lowercase letters
df['processed_review'] = df['review'].str.lower()

# Remove punctuation and special characters from the text
df['processed_review'] = df['processed_review'].apply(lambda x: re.
 ↪sub(r'[^a-z\s]', '', x))

# Remove stop words
stop_words = set(stopwords.words('english'))
df['processed_review'] = df['processed_review'].apply(lambda x: ' '.join([word␣
 ↪for word in x.split() if word not in stop_words]))

# Apply NLTK's PorterStemmer
stemmer = PorterStemmer()
df['processed_review'] = df['processed_review'].apply(lambda x: ' '.
 ↪join([stemmer.stem(word) for word in x.split()]))

# Create a bag-of-words matrix
vectorizer = CountVectorizer()
bow_matrix = vectorizer.fit_transform(df['processed_review'])
print(f"Dimensions of bag-of-words matrix: {bow_matrix.shape}")

# Create a term frequency-inverse document frequency (tf-idf) matrix
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df['processed_review'])
print(f"Dimensions of tf-idf matrix: {tfidf_matrix.shape}")
```

```
Dimensions of bag-of-words matrix: (25000, 89468)
Dimensions of tf-idf matrix: (25000, 89468)
```