DSC550_Moussadeq_Week3

March 31, 2024

DSC 550

Week 3 Assignment

Said Moussadeq

Part 1

1- Import the movie review data as a data frame and ensure that the data is loaded properly.

[12]:
```
# Import pandas library
import pandas as pd
import os
print(os.getcwd())
```

```
C:\Users\TheArchitect\Downloads
```

[13]:
```
# load the data
labeledTrainData = pd.read_csv('labeledTrainData.tsv', sep='\t')
```

[15]:
```
# View the 5 five rows
print(labeledTrainData.head(10))
```

```
        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…
5   8196_8          1  I dont know why people think this is such a ba…
6   7166_2          0  This movie could have been very good, but come…
7  10633_1          0  I watched this video at a friend's house. I'm …
8    319_1          0  A friend of mine bought this film for £1, and …
9  8713_10          1  <br /><br />This movie is full of references. …
```

2- How many of each positive and negative reviews are there?

[5]:
```
# Positive and negative reviews count
positive_count = 0
negative_count = 0

# Loop through the dataset
for sentiment in labeledTrainData['sentiment']:
    if sentiment > 0:
        positive_count += 1
    else:
        negative_count += 1

# Print the results
print("The number of positive review is :", positive_count)
print("The number of negative review is :", negative_count)
```

```
The number of positive review is : 12500
The number of negative review is : 12500
```

3- Use TextBlob to classify each movie review as positive or negative. Assume that a polarity score greater than or equal to zero is a positive sentiment and less than 0 is a negative sentiment.

[6]:
```
from textblob import TextBlob

# positive and negative reviews count
positive_count = 0
negative_count = 0

# store the sentiment in a list
predicted_sentiment = []

# Loop through the dataset
for review in labeledTrainData['review']:
    polarity = TextBlob(str(review)).polarity

  # polarity classification
    if polarity >= 0:
        sentiment = "Positive"
        positive_count += 1
    else:
        sentiment = "Negative"
        negative_count += 1

    # Add the sentiment to the list
    predicted_sentiment.append(sentiment)
```

```
# Add a new column to the dataset
labeledTrainData['predicted_sentiment'] = predicted_sentiment

# Print the results
print("The number of positive review is :", positive_count)
print("The number of negative review is :", negative_count)

# Print the first few rows of the DataFrame with sentiment column added
print("The dataset with the new column")
labeledTrainData.head(5)
```

```
The number of positive review is : 19017
The number of negative review is : 5983
The dataset with the new column
```

[6]:

	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…	

	predicted_sentiment
0	Positive
1	Positive
2	Negative
3	Positive
4	Negative

4- Check the accuracy of this model. Is this model better than random guessing?

To evaluate the model's performance, we'll align the predicted sentiment values with the true sentiment values found in our dataset. This involves transforming the model's sentiment predictions into numerical form, specifically into 0s and 1s. Subsequently, we'll perform a row-by-row comparison of these values.

[7]:
```
# intialize counters
correct_count = 0
total_count = 0

# Loop through the dataset and convert Positive sentiment to 1 and Negative to 0
for sentiment, predicted_sentiment in zip(labeledTrainData['sentiment'],␣
 ↪labeledTrainData['predicted_sentiment']):
    if predicted_sentiment == 'Positive':
        predicted_sentiment_num = 1
    else:
        predicted_sentiment_num = 0
```

```
    # Increment the number of count
    total_count += 1

    # compare the predicted value with the actual sentiment value in the dataset
    if sentiment == predicted_sentiment_num:
        correct_count += 1
# calculate the accuracy
accuracy = (correct_count/total_count)*100

print("The total count is :", total_count)
print("The matching of correct sentiment is :", correct_count)
print("The accuracy is :", accuracy, "%")
```

```
The total count is : 25000
The matching of correct sentiment is : 17131
The accuracy is : 68.524 %
```

Our next step will be to manually compute the accuracy of the model's predictions. Following this, we will validate our calculated accuracy by comparing it with the result obtained from using the accuracy_score function.

[8]:
```
from sklearn.metrics import accuracy_score
# Initialize a list to store the predicted sentiments
predicted_sentiments_sklearn = []

# Loop through the dataset
for review in labeledTrainData['review']:
    polarity = TextBlob(str(review)).polarity

    # Classify sentiment based on polarity
    if polarity >= 0:
        predicted_sentiments_sklearn.append(1)  # Positive sentiment
    else:
        predicted_sentiments_sklearn.append(0)  # Negative sentiment

# Add the predicted sentiments to the DataFrame
labeledTrainData['predicted_sentiments_sklearn'] = predicted_sentiments_sklearn

# Calculate accuracy using sklearn
accuracy = accuracy_score(labeledTrainData['sentiment'],␣
 ↪labeledTrainData['predicted_sentiments_sklearn'])

print("The accuracy is :", accuracy * 100, "%")
```

```
The accuracy is : 68.524 %
```

The accuracy of the model is 68.52% If we consider that the accuracy of random guessing is 50%, we can conclude that this model is more accurate, thus better than random guessing.

5- For up to five points extra credit, use another prebuilt text sentiment analyzer, e.g., VADER, and repeat steps (3) and (4).

[16]:
```
# Importing the pandas library
import pandas as pd

# Reading the dataset into pandas
labeledTrainData = pd.read_csv('labeledTrainData.tsv', sep='\t')
```

[21]:
```
# install vaderSentiment
!pip install vaderSentiment
```

```
Requirement already satisfied: vaderSentiment in
c:\users\thearchitect\anaconda3\lib\site-packages (3.3.2)
Requirement already satisfied: requests in
c:\users\thearchitect\anaconda3\lib\site-packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\thearchitect\anaconda3\lib\site-packages (from
requests->vaderSentiment) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\thearchitect\anaconda3\lib\site-packages (from
requests->vaderSentiment) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\thearchitect\anaconda3\lib\site-packages (from
requests->vaderSentiment) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\thearchitect\anaconda3\lib\site-packages (from
requests->vaderSentiment) (2024.2.2)
```

[24]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Initialize positive, negative, and neutral reviews count
positive_count = 0
negative_count = 0
neutral_count = 0

# Initialize lists to store the predicted sentiment
predicted_sentiment = []
predicted_sentiment_num = []  # The numerical value of the predicted sentiment
# For the assigned, a negative sentiment is 0, positive is 1
# We will use 2 for neutral sentiment

# Create a SentimentIntensityAnalyzer object
sid_obj = SentimentIntensityAnalyzer()

# Iterate through each review
for review in labeledTrainData['review']:
    # Get sentiment scores using Vader
```

```
    sentiment_dict = sid_obj.polarity_scores(review)

    # Use the compound score to decide if the review is positive, negative, or␣
 ↪neutral
    if sentiment_dict['compound'] >= 0.05:
        sentiment = "Positive"
        sentiment_num = 1
        positive_count += 1
    elif sentiment_dict['compound'] <= -0.05:
        sentiment = "Negative"
        sentiment_num = 0
        negative_count += 1
    else:
        sentiment = "Neutral"
        sentiment_num = 2
        neutral_count += 1

    # Add the sentiment to the lists
    predicted_sentiment.append(sentiment)
    predicted_sentiment_num.append(sentiment_num)


# Add the predicted sentiment and its numerical value to the dataset as new␣
 ↪columns
labeledTrainData['predicted_sentiment'] = predicted_sentiment
labeledTrainData['predicted_sentiment_num'] = predicted_sentiment_num

# Print the results
print("The number of positive reviews is:", positive_count)
print("The number of negative reviews is:", negative_count)
print("The number of neutral reviews is:", neutral_count)

# Print the first few rows of the DataFrame with the new sentiment columns
print("The dataset with the new columns:")
labeledTrainData.head(5)
```

```
The number of positive reviews is: 16507
The number of negative reviews is: 8306
The number of neutral reviews is: 187
The dataset with the new columns:
```

[24]:

	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…	

```
  predicted_sentiment  predicted_sentiment_num
0            Negative                        0
1            Positive                        1
2            Negative                        0
3            Negative                        0
4            Positive                        1
```

[11]:
```
from sklearn.metrics import accuracy_score

# Calculate accuracy using sklearn
accuracy = accuracy_score(labeledTrainData['sentiment'],␣
 ↪labeledTrainData['predicted_sentiment_num'])

print("The accuracy using VADER is :", accuracy * 100, "%")
```

```
The accuracy using VADER is : 69.016 %
```

The accuracy of the model using VADER is 69.016 %.

Part 2: Prepping Text for a Custom Model

[12]:
```
# Importing the pandas library
import pandas as pd

# Reading the dataset into pandas
labeledTrainData = pd.read_csv('labeledTrainData.tsv', sep='\t')
```

1. Convert all text to lowercase letters.

[13]:
```
# convert a text to lower case

def lower_case(text):
    text = text.lower()
    return text

# apply the function to our dataset
labeledTrainData['review_processed'] = labeledTrainData['review'].
 ↪apply(lower_case)

# Display the dataset with the new column
labeledTrainData.head(5)
```

[13]:
```
        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…

                                    review_processed
0  with all this stuff going down at the moment w…
1  \the classic war of the worlds\" by timothy hi…
2  the film starts with a manager (nicholas bell)…
3  it must be assumed that those who praised this…
4  superbly trashy and wondrously unpretentious 8…
```

2- Remove punctuation and special characters from the text.

[14]:
```
# Function to remove punctuation and special characters from the text
def remove_punct_spec_car(text):
    symbols_to_remove = [',', ';', '?', '$', '.', ':', '!', '[', ']', '(', ')',␣
 ↪'\\', '-', '"', "'"]
    # Loop through the symbols and remove them from the text
    for symbol in symbols_to_remove:
        text = text.replace(symbol, " ")
    return text

# Let's apply the function to the dataset
labeledTrainData['review_processed'] = labeledTrainData['review_processed'].
 ↪apply(remove_punct_spec_car)

# Display the dataset with the modified column
labeledTrainData.head(5)
```

[14]:
```
        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…

                                    review_processed
0  with all this stuff going down at the moment w…
1   the classic war of the worlds   by timothy hi…
2  the film starts with a manager  nicholas bell …
3  it must be assumed that those who praised this…
4  superbly trashy and wondrously unpretentious 8…
```

[15]:
```
# Let's combien the 2 functions
def  prep_text(text):
    symbols_to_remove = [',', ';', '?', '$', '.', ':', '!', '[', ']', '(', ')',␣
 ↪'\\', '-', '"', "'"]
    # Convert to lower case
```

```
    text = text.lower()
    # Remove punctuation and special characters
    for symbol in symbols_to_remove:
        text = text.replace(symbol, " ")
    return text
```

3- Remove stop words.

[16]:
```
# To remove the stop words, we need to tokenize in words
from nltk.tokenize import word_tokenize
def text_tokenize(text):
    word_tokenized = word_tokenize(text)
    return word_tokenized

labeledTrainData['review_processed'] = labeledTrainData['review_processed'].
 ↪apply(text_tokenize)

# Display the dataset with the modified column
labeledTrainData.head(5)
```

[16]:
```
        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…

                                    review_processed
0  [with, all, this, stuff, going, down, at, the,…
1  [the, classic, war, of, the, worlds, by, timot…
2  [the, film, starts, with, a, manager, nicholas…
3  [it, must, be, assumed, that, those, who, prai…
4  [superbly, trashy, and, wondrously, unpretenti…
```

[17]:
```
# Load library
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

# Download the stop words
# nltk.download('stopwords')

# Function to remove the stop words
def remove_stop_words(text):
    # Load the stop words
    stop_words = stopwords.words('english')
```

```
    # Remove the stop words
    text_modified = [word for word in text if word not in stop_words]

    return text_modified

# Apply the function to the dataset
labeledTrainData['review_processed'] = labeledTrainData['review_processed'].
 ↪apply(remove_stop_words)

# Display the dataset with the modified column
labeledTrainData.head(5)
```

[17]:

	id	sentiment	review	review_processed
0	5814_8	1	With all this stuff going down at the moment w…	[stuff, going, moment, mj, started, listening,…
1	2381_9	1	\The Classic War of the Worlds\" by Timothy Hi…	[classic, war, worlds, timothy, hines, enterta…
2	7759_3	0	The film starts with a manager (Nicholas Bell)…	[film, starts, manager, nicholas, bell, giving…
3	3630_4	0	It must be assumed that those who praised this…	[must, assumed, praised, film, greatest, filme…
4	9495_8	1	Superbly trashy and wondrously unpretentious 8…	[superbly, trashy, wondrously, unpretentious, …

4- Apply NLTK's PorterStemmer.

[18]:
```
# Load library
from nltk.stem.porter import PorterStemmer

def porter_stemmer(text):
    # Create stemmer
    porter = PorterStemmer()

    # Apply stemmer
    text_porterstemmer = [porter.stem(word) for word in text]
    return text_porterstemmer


# Apply the function
labeledTrainData['review_processed'] = labeledTrainData['review_processed'].
 ↪apply(porter_stemmer)

# Display the dataset with the modified column
labeledTrainData.head(5)
```

[18]:
```
        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…

                                    review_processed
0  [stuff, go, moment, mj, start, listen, music, …
1  [classic, war, world, timothi, hine, entertain…
2  [film, start, manag, nichola, bell, give, welc…
3  [must, assum, prais, film, greatest, film, ope…
4  [superbl, trashi, wondrous, unpretenti, 80, ex…
```

5- Create a bag-of-words matrix from your stemmed text (output from (4)), where each row is a word-count vector for a single movie review (see sections 5.3 & 6.8 in the Machine Learning with Python Cookbook). Display the dimensions of your bag-of-words matrix. The number of rows in this matrix should be the same as the number of rows in your original data frame

[19]:
```
# Loading the libraries
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

# Let's join the list of words into strings before creating the matrix
labeledTrainData['review_processed'] = labeledTrainData['review_processed'].
 ↪apply(lambda words: ' '.join(words))

# Create the bag of words feature matrix
count = CountVectorizer()
bag_of_words = count.fit_transform(labeledTrainData['review_processed'])

# Show feature matrix
bag_of_words
```

[19]:
```
<25000x52189 sparse matrix of type '<class 'numpy.int64'>'
        with 2384138 stored elements in Compressed Sparse Row format>
```

[20]:
```
# Dimensions of the bag_of_words matrix
dimension = bag_of_words.shape
print("The dimension of the bag of words is :", dimension)
```

```
The dimension of the bag of words is : (25000, 52189)
```

6- Create a term frequency-inverse document frequency (tf-idf) matrix from your stemmed text, for your movie reviews (see section 6.9 in the Machine Learning with Python Cookbook). Display the dimensions of your tf-idf matrix. These dimensions should be the same as your bag-of-words matrix.

[21]:
```
# Create a rf_idf matrix
from sklearn.feature_extraction.text import TfidfVectorizer
count_tfidf = TfidfVectorizer()
tfidf_matrix = count_tfidf.fit_transform(labeledTrainData['review_processed'])

# Show feature matrix
tfidf_matrix
```

[21]:
```
<25000x52189 sparse matrix of type '<class 'numpy.float64'>'
        with 2384138 stored elements in Compressed Sparse Row format>
```

[22]:
```
# Dimension of the tf-idf matrix
print("The dimension of the tf-idf matrix is :", tfidf_matrix.shape)
```

```
The dimension of the tf-idf matrix is : (25000, 52189)
```