Natural Language Processing-Sentiment Analysis_Completed

July 19, 2020

[1]:
```
#import required libraries
import pandas as pd
```

[4]:
```
#get the sentiment dataset
df_sentiment = pd.read_csv('imdb_labelled.
↪txt',sep='\t',names=['comment','label'])
```

[5]:
```
#view first 10 observations.
# 1 indicates positive sentiment and 0 indicate negative sentiment
df_sentiment.head(10)
```

[5]:

	comment	label
0	A very, very, very slow-moving, aimless movie …	0
1	Not sure who was more lost - the flat characte…	0
2	Attempting artiness with black & white and cle…	0
3	Very little music or anything to speak of.	0
4	The best scene in the movie was when Gerardo i…	1
5	The rest of the movie lacks art, charm, meanin…	0
6	Wasted two hours.	0
7	Saw the movie today and thought it was a good …	1
8	A bit predictable.	0
9	Loved the casting of Jimmy Buffet as the scien…	1

[6]:
```
# view more information about the setiment data using describe method
df_sentiment.describe()
```

[6]:

	label
count	748.000000
mean	0.516043
std	0.500077
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

[7]:
```
#view more info on data
df_sentiment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748 entries, 0 to 747
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   comment  748 non-null    object
 1   label    748 non-null    int64
dtypes: int64(1), object(1)
memory usage: 11.8+ KB
```

[8]:
```
# view data using group by and describe method
df_sentiment.groupby('label').describe()
```

[8]:

	comment				
	count	unique	top	freq	
label					
0	362	361	Not recommended.	2	
1	386	384	Definitely worth checking out.	2	

[9]:
```
# Verify length of the messages and also add it also as a new column (feature)
df_sentiment['length'] =df_sentiment['comment'].apply(len)
```

[10]:
```
# view first 5 messages with length
df_sentiment.head()
```

[10]:

	comment	label	length
0	A very, very, very slow-moving, aimless movie …	0	87
1	Not sure who was more lost - the flat characte…	0	99
2	Attempting artiness with black & white and cle…	0	188
3	Very little music or anything to speak of.	0	44
4	The best scene in the movie was when Gerardo i…	1	108

[11]:
```
#view first
df_sentiment[df_sentiment['length']>50]['comment'].iloc[0]
```

[11]:
```
'A very, very, very slow-moving, aimless movie about a distressed, drifting
young man.  '
```

[12]:
```
# start text processing with vectorizer
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
```

[13]:
```
# define a function to get rid of stopwords present in the messages
def message_text_process(mess):
```

```
    # Check characters to see if there are punctuations
    no_punctuation = [char for char in mess if char not in string.punctuation]
    # now form the sentence.
    no_punctuation = ''.join(no_punctuation)
    # Now eliminate any stopwords
    return [word for word in no_punctuation.split() if word.lower() not in␣
 ↪stopwords.words('english')]
```

[14]:
```
# bag of words by applying the function and fit the data (comment) into it
import string
from nltk.corpus import stopwords
bag_of_words = CountVectorizer(analyzer=message_text_process).
 ↪fit(df_sentiment['comment'])
```

[15]:
```
# apply transform method for the bag of words
comment_bagofwords = bag_of_words.transform(df_sentiment['comment'])
```

[16]:
```
# apply tfidf transformer and fit the bag of words into it (transformed version)
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer().fit(comment_bagofwords)
```

[18]:
```
# print shape of the tfidf
comment_tfidf = tfidf_transformer.transform(comment_bagofwords)
comment_tfidf.shape
```

[18]:
```
(748, 3259)
```

[19]:
```
#choose naive Bayes model to detect the spam and fit the tfidf data into it
from sklearn.naive_bayes import MultinomialNB
sentiment_detection_model = MultinomialNB().
 ↪fit(comment_tfidf,df_sentiment['label'])
```

[21]:
```
# check model for the predicted  and expected value say for comment# 1 and␣
 ↪comment#5
comment = df_sentiment['comment'][4]
bag_of_words_for_comment = bag_of_words.transform([comment])
tfidf = tfidf_transformer.transform(bag_of_words_for_comment)

print('predicted sentiment label ', sentiment_detection_model.predict(tfidf)[0])
print('expected sentiment label', df_sentiment.label[4])
```

```
predicted sentiment label  1
expected sentiment label 1
```

[ ]: