Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2023-24

Class:	TE	**Semester:**	VI
Course Code:	CSL601	**Course Name:**	Data Analytics and Visualization

Name of Student:	Ojasi Prashant Prabhu
Roll No.:	43
Experiment No.:	6
Title of the Experiment:	Text analytics:Implementation of Spam filter/Sentiment analysis in Python/R
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Ms Bhavika Gharat

Signature :

Date :

EXPERIMENT NO 6

Aim: Implementation of Sentiment Analysis

Objective:- To understand the use of various sentiment Analysis techniques by implementing them

Description:

Sentiment Analysis: Sentiment Analysis is a text analysis technique that allows companies to make sense of qualitative data. By detecting positive and negative sentiment in text data, such as tweets, product reviews, and support tickets, you can understand how customers feel about your brand, product, or service, and gain insights that lead to data-driven decisions

Sentiment Analysis deals with analyzing emotions and the perspective of a speaker or an author from a given piece of text. "Sentiment analysis or opinion mining refers to the appliance of language process, linguistics, and text analytics to spot and extract subjective information in supply materials". This field of technology deals with analyzing and predicting the hidden information keep within the text. This hidden information gives valuable insights regarding user's intentions, style and odds. Sentiment Analysis specializes in categorizing the text at the extent of subjective and objective nature. Judgement indicates that the text bears opinion content where's perspicacity indicates that the text is while not opinion content

Some examples-

1. Subjective- This motion picture by tom cruise and Angelina jolie is great. (This sentence has an opinion, it talks regarding the motion picture and also the writer's emotions regarding same "great" and thence its subjective

2. Objective- This motion picture stars tom cruise and Angelina.

(This sentence may be a reality, general information instead of an opinion or a read of some individual and thence its objective)

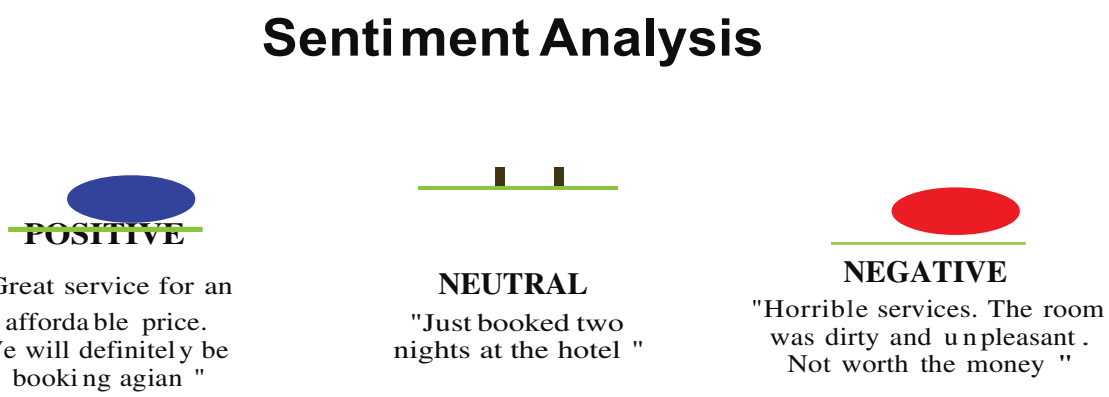
The subjective text may be additional categorized into three broad classes supported the emotions expressed within the text.

1. Positive- I like to look at Star series.

2. Negative- The movie was awful.

3. Neutral- I typically get hungry by evening. (This sentence has users views, emotions hence it's subjective however because it doesn't have any positive or negative polarity therefore it's neutral).

Example:



Method 1: Using Positive and Negative Word Count – With Normalization for Calculating Sentiment Score

In this method, we will calculate the Sentiment Scores by classifying and counting the Negative and Positive words from the given text and taking the ratio of the difference of Positive and Negative Word Counts and Total Word Count. We will be using the Amazon Cell Phone Reviews dataset from Kaggle.

To calculate the sentiment, here we will use the formula:

Let's first import the Libraries

```
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import re
```

Now, we will import the dataset. Out of all the columns present in this dataset, we will use the 'body' column which contains the cell phone reviews.

```
df = pd.read_csv('20191226-reviews.csv' , usecols=['body'])
```

Next, we will create an instance of WordNetLemmatizer, which we will be using in the next step. Also, we will call the Stop Words from the NLTK library.

```
lemma = WordNetLemmatizer()
stop_words = stopwords.words('english')
```

Next, we will define a function to preprocess the text. In this function, first, we converted the input string to lowercase. Then, we extracted only the letters from this string. This task is performed using Regular Expressions. Next, we tokenized the string. After tokenizing, we filtered the text of stop words. At last, we lemmatized the remaining words and returned these words.

```
def text_prep(x):
    corp = str(x).lower()
    corp = re.sub('[^a-zA-Z]+',' ', corp).strip()
    tokens = word_tokenize(corp)
    words = [t for t in tokens if t not in stop_words]
    lemmatize = [lemma.lemmatize(w) for w in words]

    return lemmatize
```

Now, we will apply this function to every text in the **'body'** column.

```
preprocess_tag = [text_prep(i) for i in df['body']]
df["preprocess_txt"] = preprocess_tag
```

Next, we will calculate the number of resultant words for each text. This will be helpful later when we calculate the Sentiment Score.

```
df['total_len'] = df['preprocess_txt'].map(lambda x: len(x))
```

In the next step, we defined the generic Negative Words and Positive Words list. For this, we have taken the **positive-text.txt** and **negative-text.txt** files or generally known as Opinion Lexicon by the respective authors(s).

Next, we will make a count of positive and negative words for the text. For this, we will count all the words in **preprocess_txt** that are present in positive words list **pos_words,** and similarly will count all the words in **preprocess_txt** that are present in the negative words list **neg_words.** We will add these values into the main Pandas DataFrame.

```
num_pos = df['preprocess_txt'].map(lambda x: len([i for i in x if i in pos_words]))
df['pos_count'] = num_pos
num_neg = df['preprocess_txt'].map(lambda x: len([i for i in x if i in neg_words]))
df['neg_count'] = num_neg
```

Finally, we will calculate the sentiment. We will create a **'sentiment'** column and perform the calculation for each text.

```
df['sentiment'] = round((df['pos_count'] - df['neg_count']) / df['total_len'], 2)
```

Putting it all together:

```
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
from nltk.stem import WordNetLemmatizer
import re
df = pd.read_csv('20191226-reviews.csv' , usecols=['body'])
lemma = WordNetLemmatizer()
stop_words = stopwords.words('english')
def text_prep(x: str) -> list:
    corp = str(x).lower()
    corp = re.sub('[/\a-zA-Z]+',' ', corp).strip()
    tokens = word_tokenize( corp)
    words = [t for t in tokens if t not in stop_words]
    lemmatize = [lemma.lemmatize(w) for w in words]
    return lemmatize
preprocess_tag = [text_prep(i) for i in df['body']]
df["preprocess_txt"] = preprocess_tag
df['total_len'] = df['preprocess_txt'].map(lambda x: len(x))
file = open('negative-words.txt' , 'r')
neg_words = file.read().split()
file = open('positive-words.txt' , 'r')
pos_words = file.read().split()
num_pos = df['preprocess_txt'].map(lambda x: len([i for i in x if i in pos_words]))
df['pos_count'] = num_pos
num_neg = df['preprocess_txt'].map(lambda x: len([i for i in x if i in neg_words]))
df['neg_count'] = num_neg
df['sentiment'] = round((df['pos_count'] - df['neg_count']) / df['total_len'], 2)
df.head()
```

On executing this code, we get:

	bod(y	1we1>r<>C€-ss_bd	total_len	J>DS_<><>unt	ne<;11_count	sentiment
0	Ih"dth:e Samsung A!JOO fur a>'iihile whid>is abs...	[:oamsung, ">'iihile, at>solurte, <foo, doo, read, re...	162	1&	11!	-0,01
1	**Due:to a software issue be1we-e.n lNotia and Spfi...**	Idu.,, s<>flware, issu», n.ol:ia, sp<int, phone, t...	67	&	3	0.07
2	**This is a great, reliable ph.one. I .also pu1ch:a...**	(great, reliable, J>hone, "lso, J>UJmased, Ph<>n...	e.s	10	4	O.OI!
3	I lov" the P1101>e an.d all. beosus>e I really did...	[lov.,, Phon:e, really, need, """· exJ><>d, p<i>oe...	41	3	0	0.07
4	The Phone has been great fur eveiy J>UJjl<l'2 it ...	[11hone, gJeat. every, purjl<l5<>, ofiel, exceJ>I, ...	56	5	3	0.04

Method 2: Using Positive and Negative Word Counts – With Semi Normalization to calculate Sentiment Score

In this method, we calculate the sentiment score by evaluating the ratio of Count of Positive Words and Count of Negative Words + 1. Since there is no difference of values involved, the sentiment value will always be more than 0. Also, adding 1 in the denominator would save from Zero Division Error. Let's start with the implementation. The implementation code will remain the same till the Negative and Positive Word Count from the previous part with a difference that this time we don't need the total word count, thus will be omitting that part.

```
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import re
df = pd.read_csv('20191226-reviews.csv' , usecols=['body'])
lemma = WordNetLemmatizer()
stop_words = stopwords.words('english')
def text_prep(x: str) -> list:
    corp = str(x).lower()
    corp = re.sub('[/\a-zA-Z]+',' ', corp).strip()
    tokens = word_tokenize( corp)
    words = [t for t in tokens if t not in stop_words]
    lemmatize = [lemma.lemmatize(w) for w in words]
    return lemmatize
preprocess_tag = [text_prep(i) for i in dfl'body']]
dfl"preprocess_txt"] = preprocess_tag
file = open('negative-words.txt' , 'r')
neg_words = file.read().split()
file = open('positive-words.txt' , 'r')
pos_words = file.read().split()
num_pos = dfl'preprocess_txt'].map(lambda x: len([i for i in x if i in pos_words]))
dfl'pos_count'] = num_pos
num_neg = dfl'preprocess_txt'].map(lambda x: len([i for i in x if i in neg_words]))
dfl'neg_count'] = num_neg
```

Next, we will apply the formula and add the formulated value into the main DataFrame.

```
dfl'sentiment'] = round(dfl'pos_count'] / (dfl'neg_count']+ 1), 2)
```

Putting it all together.

```
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import re
df = pd.read_csv('20191226-reviews.csv' , usecols=['body'])
lemma = WordNetLemmatizer()
stop_words = stopwords.words('english')
def text_prep(x: str) -> list:
    corp = str(x).lower()
    corp = re.sub('[/\a-zA-Z]+',' ', corp).strip()
    tokens = word_tokenize( corp)
    words = [t for t in tokens if t not in stop_words]
    lemmatize = [lemma.lemmatize(w) for w in words]
    return lemmatize
preprocess_tag = [text_prep(i) for i in dfl'body']]
dfl"preprocess_txt"] = preprocess_tag
```

```
file = open('negative-words.txt' , 'r')
neg_words = file.read().split()
file = open('positive-words.txt' , 'r')
pos_words = file.read().split()
num_pos = dfl'preprocess_txt'].map(lambda x: len([i for i in x if i in pos_words]))
dfl'pos_count'] = num_pos
num_neg = dfl'preprocess_txt'].map(lambda x: len([i for i in x if i in neg_words]))
dfl'neg_count'] = num_neg
dfl'sentiment'] = round(dfl'pos_count'] / (dfl'neg_count']+ 1), 2)
df.head()
```

On executing this code, we get:

		re1Process_txt	111os._count	neg1_coun.t	sentime;nt
0	I had the Samsung A600 f0< awhile w:hiCh is abs...	[samsung, awhile, absolute, dOO, dOO, rse.d, re...	1&	1Q	0.00
1	0-u:e to a software issue between Nol<ia and S:pri...	[d'tr,e, software, issue, nc ia, SJllint, p!lo:n.e, t...	&	3	2.00
2	This is a great, relia'ble ne. I also purCha...	[great, reliable, Ph<>ne, also, purdlaS'Eid, phon..	10	4	2.00
3.	I love the Ph<>n.E and all, Ilecs use I really dlidl...	[love, phone, really, need, ane, •.:•lpect, Jlli•ce...	3	0	3.00
4	The phone has been great f0< evert purpooe it ...	[p!ho:ne, great, everf, pu pc>S'e, offe<, eJ<oe,pt, ...	5	3	1.25

Conclusion-

Techniques used for sentimental analysis are

Lexicons: Pre-built dictionaries with sentiment scores assigned to words.
Machine Learning: Training models on labeled data to classify sentiment.
Deep Learning: RNNs and LSTMs capture context for nuanced analysis.
Hybrid Approaches: Combining techniques for improved accuracy.