CS335

Lecturer : Dr. Dinesh Kumar

B.Tech Year – III, Vth Sem

Academic Year (2022-23)

IIITV-ICD

Sentiment analysis.

Machine Learning Mini Project report.

Group member : Hari Jarupla (202011024)

Description :

It is a Natural Language Processing Problem where Sentiment Analysis is done by Classifying the Positive tweets from negative tweets by machine learning models for classification, text mining, text analysis, data analysis and data visualization.

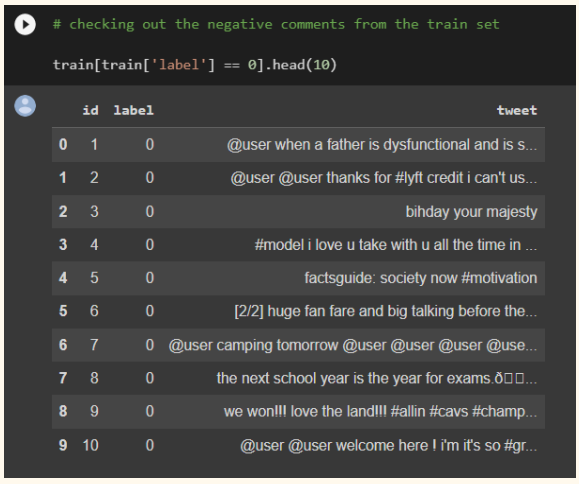
Introduction :

Natural Language Processing (NLP) is a hotbed of research in data science these days and one of the most common applications of NLP is sentiment analysis. From opinion polls to creating entire marketing strategies, this domain has completely reshaped the way businesses work, which is why this is an area every data scientist must be familiar with.

Thousands of text documents can be processed for sentiment (and other features including named entities, topics, themes, etc.) in seconds, compared to the hours it would take a team of people to manually complete the same task.

The objective of this task is to detect hate speech in tweets. For the sake of simplicity, we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets.

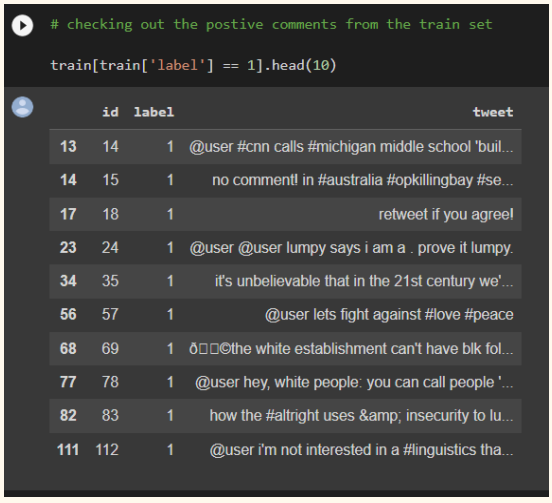
Formally, given a training sample of tweets and labels, where label '1' denotes the tweet is racist/sexist and label '0' denotes the tweet is not racist/sexist, your objective is to predict the labels on the given test dataset.



```
# checking out the negative comments from the train set

train[train['label'] == 0].head(10)
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
5	6	0	[2/2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...
7	8	0	the next school year is the year for exams.ð□□...
8	9	0	we won!!! love the land!!! #allin #cavs #champ...
9	10	0	@user @user welcome here ! i'm it's so #gr...



```
# checking out the postive comments from the train set

train[train['label'] == 1].head(10)
```

	id	label	tweet
13	14	1	@user #cnn calls #michigan middle school 'buil...
14	15	1	no comment! in #australia #opkillingbay #se...
17	18	1	retweet if you agree!
23	24	1	@user @user lumpy says i am a . prove it lumpy.
34	35	1	it's unbelievable that in the 21st century we'...
56	57	1	@user lets fight against #love #peace
68	69	1	ð□□©the white establishment can't have blk fol...
77	78	1	@user hey, white people: you can call people '...
82	83	1	how the #altright uses & insecurity to lu...
111	112	1	@user i'm not interested in a #linguistics tha...

Note: The evaluation metric from this practice problem is F1-Score.

Problem definition :

To be able to accurately predict whether a tweet is a positive statement or a negative (racist/swear/harassment) statement.

3. The learning system (Algorithm)

We will do so by following a sequence of steps needed to solve a general sentiment analysis problem. We will start with preprocessing and cleaning of the raw text of the tweets. Then we will explore the cleaned text and try to get some intuition about the context of the tweets. After that, we will extract numerical features from the data and finally use these feature sets to train models and identify the sentiments of the tweets.

a. Preprocessing of text.

The Training Dataset is of **31962** tweets and the testing dataset is of **49159** tweets.

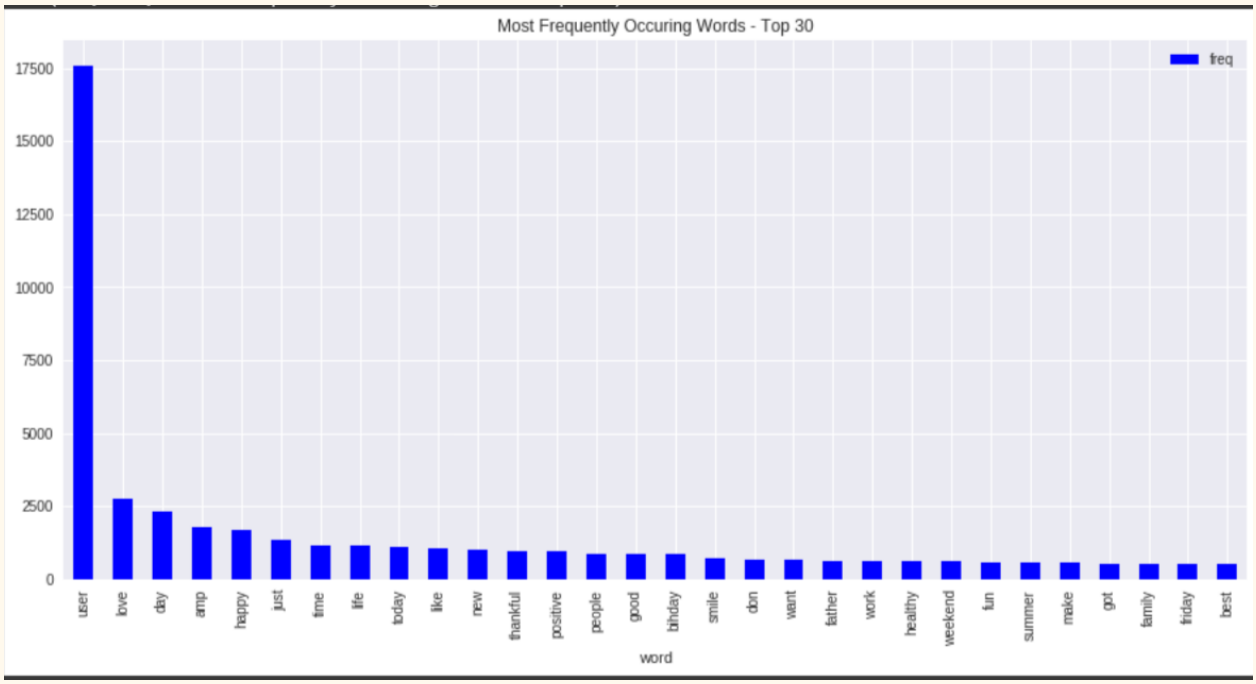
If the data is arranged in a structured format then it becomes easier to find the right information.

The preprocessing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. If we skip this step then there is a higher chance that you are working with noisy and inconsistent data. The objective of this step is to clean noise that is less relevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms which don't carry much weightage in context to the text.

➔ **Countvectorizer**

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

Countvectorizer is a tool by sklearn used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.



The above output can be seen for the understanding,but in real running it will produce a matrix along with word and frequency

➔ **Remove hashtag**

```
[ ]  # collecting the hashtags

     def hashtag_extract(x):
         hashtags = []

         for i in x:
             ht = re.findall(r"#(\w+)", i)
             hashtags.append(ht)

         return hashtags
```

```
     # extracting hashtags from non racist/sexist tweets
     HT_regular = hashtag_extract(train['tweet'][train['label'] == 0])

     # extracting hashtags from racist/sexist tweets
     HT_negative = hashtag_extract(train['tweet'][train['label'] == 1])

     # unnesting list
     HT_regular = sum(HT_regular,[])
     HT_negative = sum(HT_negative,[])
```

```
[ ]  a = nltk.FreqDist(HT_regular)
     d = pd.DataFrame({'Hashtag': list(a.keys()),
                       'Count': list(a.values())})

     # selecting top 20 most frequent hashtags
     d = d.nlargest(columns="Count", n = 20)
     plt.figure(figsize=(16,5))
     ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
     ax.set(ylabel = 'Count')
     plt.show()
```

➔ **Tokenizing**

```
# tokenizing the words present in the training set
tokenized_tweet = train['tweet'].apply(lambda x: x.split())

# importing gensim
import gensim

# creating a word to vector model
model_w2v = gensim.models.Word2Vec(
            tokenized_tweet,
            size=200, # desired no. of features/independent variables
            window=5, # context window size
            min_count=2,
            sg = 1, # 1 for skip-gram model
            hs = 0,
            negative = 10, # for negative sampling
            workers= 2, # no.of cores
            seed = 34)

model_w2v.train(tokenized_tweet, total_examples= len(train['tweet']), epochs=20)
```

```
(6109793, 8411580)
```

➔ Stopwords

```
# removing unwanted patterns from the data


import re

import nltk
```

```
nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer
```

➔ Standard scaler

```
➔ # standardization
➔
➔ from sklearn.preprocessing import StandardScaler
➔
➔ sc = StandardScaler()
➔
➔ x_train = sc.fit_transform(x_train)
➔ x_valid = sc.transform(x_valid)
➔ x_test = sc.transform(x_test)
```

b. Model algorithms and Result analysis

Terminologies :

➔ **Training accuracy :**

- ◆ Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

➔ **Validation accuracy :**

- ◆ The test (or testing) accuracy often refers to the validation accuracy, that is, the accuracy you calculate on the data set you do not use for training, but you use (during the training process) for validating (or "testing").
  - $$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$
  - *TP* : True Positives
  - *FP* : False Positives
  - *TN* : True Negatives
  - *FN* : False Negatives

➔ **F1-score :**

- ◆ F1 score is defined as the harmonic mean between precision and recall. It is used as a statistical measure to rate performance. In other words, an F1-score (from 0 to 9, 0 being lowest and 9 being the highest) is a mean of an individual's performance, based on two factors i.e. precision and recall.
  - $$F = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

➔ Random Forest

- ***Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset***

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

model = RandomForestClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))
```

-
- ***Output :***

```
Training Accuracy : 0.9946602144257645
Validation Accuracy : 0.9501939682142411
F1 score : 0.6004016064257027
[[7294  138]
 [ 260  299]]
```

➔ Decision Tree

- Decision tree is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

- **Output :**

```
Training Accuracy : 0.9991656585040257
Validation Accuracy : 0.9326742585408585
f1 score : 0.5393835616438356
[[7138  294]
 [ 244  315]]
```

➔ Logistic regression

- Logistic Regression is a **classification algorithm that is used to predict the probability of a categorical dependent variable**. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

- **Output :**

```
Training Accuracy : 0.984773267698469
Validation Accuracy : 0.9410586910274058
f1 score : 0.5915004336513443
[[7179  253]
 [ 218  341]]
```

Reference :

Yi, S. and Liu, X.

Yi, S. and Liu, X. (2020) "Machine learning based customer sentiment analysis for recommending shoppers, shops based on customers' review", *Complex & Intelligent Systems*, 6(3), pp. 621-634. doi: 10.1007/s40747-020-00155-2.

Sentiment analysis - monkeylearn

Sentiment analysis - monkeylearn(2022).

Available at: https://monkeylearn.com/sentiment-analysis/

(Accessed: 4 November 2022).