UML501 – Machine Learning Lab

TWITTER SENTIMENT ANALYSIS

UML501 Machine Learning Project Report

Submitted by:

102003362 Nikunj Bansal
102003366 Maanya Jain

BE THIRD Year,

COE Group No: 3CO14

Submitted to: Dr. Suchita Sharma

Computer Science and Engineering Department

ABSTRACT

This project addresses the problem of sentiment analysis in twitter, that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with around 400 million registered users out of which 206 million are active users, generating nearly 500 million tweets per day. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analyzing the sentiments expressed in the tweets. Analyzing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream.

INTRODUCTION

MOTIVATION

We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analyzing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm's stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favorable response and in which a negative response (since twitter allows us to download stream of geo-tagged tweets for particular locations. If firms can get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis.

DOMAIN INTRODUCTION

This project of analyzing sentiments of tweets comes under the domain of "Pattern Classification" and "Data Mining". Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering "useful" patterns in large set of data, either automatically (unsupervised) or semi-automatically (supervised). The project would heavily rely on techniques of "Natural Language Processing" in extracting significant patterns and features from the large data set of tweets and on "Machine Learning" techniques for accurately classifying individual un-labelled data samples (tweets) according to whichever pattern model best describes them.

The features that can be used for modeling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word "excellent" has a strong positive connotation while the word "evil" possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analyzing the frequency distribution of these parts of speech (ether individually or collectively with some other part of speech) in a particular class of labeled tweets. Twitter based features are more informal and relate with how people

express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, question marks, presence of URL in tweets, exclamation marks, internet emoticons and internet shorthand/slangs.

TECHNOLOGIES USED

Programming Language:

- Python

Technologies, libraries and frameworks:

- Python:
  - Numpy
  - Pandas
  - NLTK
  - String
  - Sklern

IDE:

- Jupyter Notebook

Versioning Control:

- Git and Github

DATASET DESCRIPTION

Dataset is downloaded from Kaggle website. Data is in the form of rows and columns which contains about 10980 rows and 12 columns. The dataset contains many attributes like “tweet_id”, “airline_sentiment”, “airline”,“airline_sentiment_goal”, “name”, “negativereason_gold”, “retweet_count”, “text,tweet_coord”, “tweet_created”, “tweet_location”, “user_timezone”. From all the mentioned features, the most useful features for our study were “text” which contains the actual tweet done by a twitter user and the “airline_sentiment” which contains the labels of tweets in three classes according to the sentiments expressed/observed in the tweets: positive, negative and neutral.

Original Dataset Link:
https://www.kaggle.com/datasets/maanyajain/twitter-sentiment-analysis

FEATURE EXTRACTION

Now that we have arrived at our training dataset we need to extract useful features from it which can be used in the process of classification. Some text formatting techniques which will aid us in feature extraction:

• Extraction of desired columns: “text” and “airline_sentiment” are the useful columns that are extracted from the dataset.

```
df_train=df_train[['text','airline_sentiment']]
df_train.head()
```

	text	airline_sentiment
0	@SouthwestAir I am scheduled for the morning, ...	negative
1	@SouthwestAir seeing your workers time in and ...	positive
2	@united Flew ORD to Miami and back and had gr...	positive
3	@SouthwestAir @dultch97 that's horse radish 😤 🐴	negative
4	@united so our flight into ORD was delayed bec...	negative

• Tokenization: It is the process of breaking a stream of text up into words, symbols and other meaningful elements called “tokens”. Tokens can be separated by whitespace characters and/or punctuation characters. It is done so that we can look at tokens as individual components that make up a tweet.

```
from nltk.tokenize import word_tokenize

tweets_train=[]
for i in range(len(training_documents)):
    tweets_train.append([word_tokenize(training_documents[i][0]),training_documents[i][1]])
```

• URL's and user references (identified by tokens “http” and “@”) are removed if we are interested in only analyzing the text of the tweet.

• Punctuation marks and digits/numerals may be removed if for example we wish to compare the tweet to a list of English words.

```
import string
from nltk.corpus import stopwords
stops=set(stopwords.words("english"))
punctuations=list(string.punctuation)
stops.update(punctuations)
stops, string.punctuation
```

• Lowercase Conversion: Tweet may be normalized by converting it to lowercase which makes it’s comparison with an English dictionary easier.

• Lemmatization: In computational linguistics, lemmatization is the algorithmic process of determining the lemma of a word based on its intended meaning.

```
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag
lemmatizer=WordNetLemmatizer()
```

```
def clean_tweets(words):
    output_words=[]
    for w in words:
        if w.isalpha():
            if w.lower() not in stops:
                pos=pos_tag([w])
                clean_word=lemmatizer.lemmatize(w,pos=get_simple_pos(pos[0][1]))
                output_words.append(clean_word.lower())
    return output_words
```

• Stop-words removal: Stop words are class of some extremely common words which hold no additional information when used in a text and are thus claimed to be useless. Examples include “a”, “an”, “the”, “he”, “she”, “by”, “on”, etc. It is sometimes convenient to remove these words because they hold no additional information since they are used almost equally in all classes of text.

Identifying StopWords Along with punctuations

```
import string
from nltk.corpus import stopwords
stops=set(stopwords.words("english"))
punctuations=list(string.punctuation)
stops.update(punctuations)
stops, string.punctuation
```

• Parts-of-Speech Tagging: POS-Tagging is the process of assigning a tag to each word in the sentence as to which grammatical part of speech that word belongs to, i.e. noun, verb, adjective, adverb, coordinating conjunction etc.

Part Of Speech using NLTK

```
from nltk.corpus import wordnet
```

```
def get_simple_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

SENTIMENT ANALYSIS

The classification algorithms from the Scikit-learn (Sklearn) library are now applied to fit the training data and predict the output for the testing data. Below are the classification algorithms that we have applied on the training data to predict the result for testing data:

- **Support Vector Machine**
  The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM (Support Vector Machine)

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
svc=SVC()
svc.fit(x_train_tweets,y_train_tweets)
y_test_pred=svc.predict(x_test_tweets)
print("Classification Report:")
print(classification_report(y_test_tweets,y_test_pred))
print("Confusion Matrix: ")
print(confusion_matrix(y_test_tweets,y_test_pred))
print("Accuracy Score: ")
print(accuracy_score(y_test_tweets,y_test_pred)*100,"%",sep=" ")
```

```
Classification Report:
              precision    recall  f1-score   support

    negative       0.81      0.92      0.86       687
     neutral       0.64      0.49      0.55       245
    positive       0.70      0.55      0.62       166

    accuracy                           0.77      1098
   macro avg       0.72      0.65      0.68      1098
weighted avg       0.76      0.77      0.76      1098

Confusion Matrix:
[[631  36  20]
 [106 120  19]
 [ 42  32  92]]
Accuracy Score:
76.775956284153 %
```

```
svc.fit(x_train_features,y_train)
y_pred_svc=svc.predict(x_test_features)
```

```
df=pd.DataFrame(y_pred_svc)
df.to_csv('predictions_tweets_svm.csv',index=False,header=False)
```

- **Random Forest Classifier**

  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.

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
rfc=RandomForestClassifier()
rfc.fit(x_train_tweets,y_train_tweets)
y_test_pred=rfc.predict(x_test_tweets)
print("Classification Report:")
print(classification_report(y_test_tweets,y_test_pred))
print("Confusion Matrix: ")
print(confusion_matrix(y_test_tweets,y_test_pred))
print("Accuracy Score: ")
print(accuracy_score(y_test_tweets,y_test_pred)*100,"%",sep=" ")
```

```
Classification Report:
              precision    recall  f1-score   support

    negative       0.82      0.87      0.84       687
     neutral       0.59      0.50      0.54       245
    positive       0.66      0.63      0.64       166

    accuracy                           0.75      1098
   macro avg       0.69      0.67      0.68      1098
weighted avg       0.74      0.75      0.75      1098

Confusion Matrix:
[[599  56  32]
 [100 122  23]
 [ 32  29 105]]
Accuracy Score:
75.22768670309654 %
```

```
rfc.fit(x_train_features,y_train)
y_pred_rfc=rfc.predict(x_test_features)
```

```
df=pd.DataFrame(y_pred_rfc)
df.to_csv('predictions_tweets_rfc.csv',index=False,header=False)
```

- **Multinomial Naive Bayes Classifier**

  Multinomial Naive Bayes algorithm is a probabilistic learning method. The algorithm is based on the Bayes theorem and predicts the tag of a text. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

Multinomial Naive Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
mnv=MultinomialNB(alpha=1)
mnv.fit(x_train_tweets,y_train_tweets)
y_test_pred=mnv.predict(x_test_tweets)
print("Classification Report:")
print(classification_report(y_test_tweets,y_test_pred))
print("Confusion Matrix: ")
print(confusion_matrix(y_test_tweets,y_test_pred))
print("Accuracy Score: ")
print(accuracy_score(y_test_tweets,y_test_pred)*100,"%",sep=" ")
```

```
Classification Report:
              precision    recall  f1-score   support

    negative       0.84      0.87      0.85       687
     neutral       0.60      0.53      0.56       245
    positive       0.65      0.65      0.65       166

    accuracy                           0.76      1098
   macro avg       0.70      0.68      0.69      1098
weighted avg       0.76      0.76      0.76      1098

Confusion Matrix:
[[599  58  30]
 [ 88 130  27]
 [ 29  29 108]]
Accuracy Score:
76.22950819672131 %
```

```
mnv.fit(x_train_features,y_train)
y_pred_mnv=mnv.predict(x_test_features)
```

```
df=pd.DataFrame(y_pred_mnv)
df.to_csv('Predictions_tweets_mnv.csv',index=False,header=False)
```

- **Decision Tree Classifier**

  A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks.

Decision Tree Classifier

```
from sklearn.datasets import make_classification
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from sklearn import tree
dt = tree.DecisionTreeClassifier()
dt.fit(x_train_tweets,y_train_tweets)
y_test_pred=dt.predict(x_test_tweets)
print("Classification Report:")
print(classification_report(y_test_tweets,y_test_pred))
print("Confusion Matrix: ")
print(confusion_matrix(y_test_tweets,y_test_pred))
print("Accuracy Score: ")
print(accuracy_score(y_test_tweets,y_test_pred)*100,"%",sep=" ")
```

```
Classification Report:
              precision    recall  f1-score   support

    negative       0.79      0.79      0.79       687
     neutral       0.48      0.47      0.47       245
    positive       0.57      0.60      0.58       166

    accuracy                           0.69      1098
   macro avg       0.61      0.62      0.61      1098
weighted avg       0.69      0.69      0.69      1098

Confusion Matrix:
[[542  95  50]
 [106 114  25]
 [ 38  29  99]]
Accuracy Score:
68.76138433515483 %
```

```
dt.fit(x_train_features,y_train)
y_pred_dt=dt.predict(x_test_features)
```

```
df=pd.DataFrame(y_pred_dt)
df.to_csv('Predictions_tweets_dt.csv',index=False,header=False)
```

RESULT AND CONCLUSION

For all the four classification algorithms that were applied on the training dataset to predict the output for the testing data the accuracy score was calculated. Among all the four algorithms that were used, Support Vector Machine (SVM) was giving the best accuracy score of about 76.7%.

Among all the 4 Classifiers, which are applied SVM predicts Result with higher Accuracy Score

So, following is the Result of Predicted Airline Sentiment of Testing Data through SVM

```
import csv
i=0
v=open("Predictions_tweets_svm.csv")
r = csv.reader(v)
for item in r:
    item.insert(0,tweets_test[i])
    i+=1
    print(item)
```

```
['americanair car gng dfw pulled ago icy road aa since ca reach arpt wat', 'negative']
['americanair plane land identical bad condition grk accord metars', 'negative']
['southwestair ca believe many pay customer left high dry reason flight cancelled flightlations monday bdl wow', 'negative']
['usairways legitimately say would rather driven cross country flown us airways', 'negative']
['americanair still response aa great job guy', 'positive']
['united developer fly tmrw morn min layover earlier flight layover move', 'negative']
['usairways hello anyone', 'negative']
['usairways husainhaqqani husain u shld protest well one ur party member rehman malik delayed pia flight hour', 'negative']
['usairways likely flightaware say plane still durango depart', 'negative']
['americanair even give option hold say line busy plz try late flightr', 'negative']
['united announcement pre boarding address mobility disability require travel lot stuff preboard', 'negative']
['usairways really embarrass ask complimentary detailed http amp argue', 'negative']
['southwestair passport time trip could still fly photo id thingsishouldknow ifeeldumb', 'negative']
['americanair delayed bag friend lisa pafe get bag day costa rica issue update system', 'negative']
['southwestair see travel compete unused fund expiration date hidden fine print never saw', 'negative']
['usairways awesome doors close minute flight leaf minute plane get wth', 'negative']
['united flew united last month experience awesome', 'negative']
['jetblue accepting apple pay mobile enterprise http', 'neutral']
['united cab ride dfw love get bag reimburse', 'negative']
['jetblue ill call morning upset right', 'negative']
['southwestair aarp appreciate tweet back unexpected', 'positive']
['united status flight', 'neutral']
['americanair literally stop allow people wait line customer service incredible bad customer service', 'negative']
['americanair really want get home tonight preferably please stop delay plane americanairlines', 'negative']
['usairways hold reservation desk hour help', 'negative']
['jetblue flight flight plane visible snack crumb seat upon boarding', 'negative']
['americanair nothing mother nature like poor commutation', 'negative']
```