CSC 5354
EL AMRI Ali
72241

Sentiment Analysis

OVERVIEW

- The objective of this project is to apply classification techniques to sentiment analysis and to become familiar with Python and NLTK as a tool for processing text and performing natural language processing text.

INSTALLATION

1. Browse to the repository link and either download as a zip or clone onto your machine.
2. Download the IMDb data set and place somewhere on your computer.
3. Extract the content of the zipped folder into these paths respectively:
    a. test/pos => imdb\testing\data\positive
    b. test/neg => imdb\testing\data\negative
    c. train/pos => imdb\training\data\positive
    d. train/pos => imdb\training\data\positive
4. Download the sentube dataset provided in the project's description or via this link
5. Extract its content to into these paths respectively:
    a. automobiles_EN => sentube\data\automobiles
    b. tablets_EN => sentube\data\tablets
6. Browse to the Kaggle US airline data set and hit download
7. Extract the CSV file to twitter\training\data

RECAP

We have downloaded the sentube data that contains youtube comments on videos related to automobiles and tablets that will serve as data to be analysed.

The very large IMDb movie reviews dataset and the tweets about US airlines will serve as foundation of our classifiers to be trained on.

DEPENDENCIES

The project uses python 3.6.5 and requires somes libraries to be installed, this can be done through pip that we will use to install the following:

- NLTK
- SciKit
- Numpy
- Scipy

You can install all the above dependencies by running:

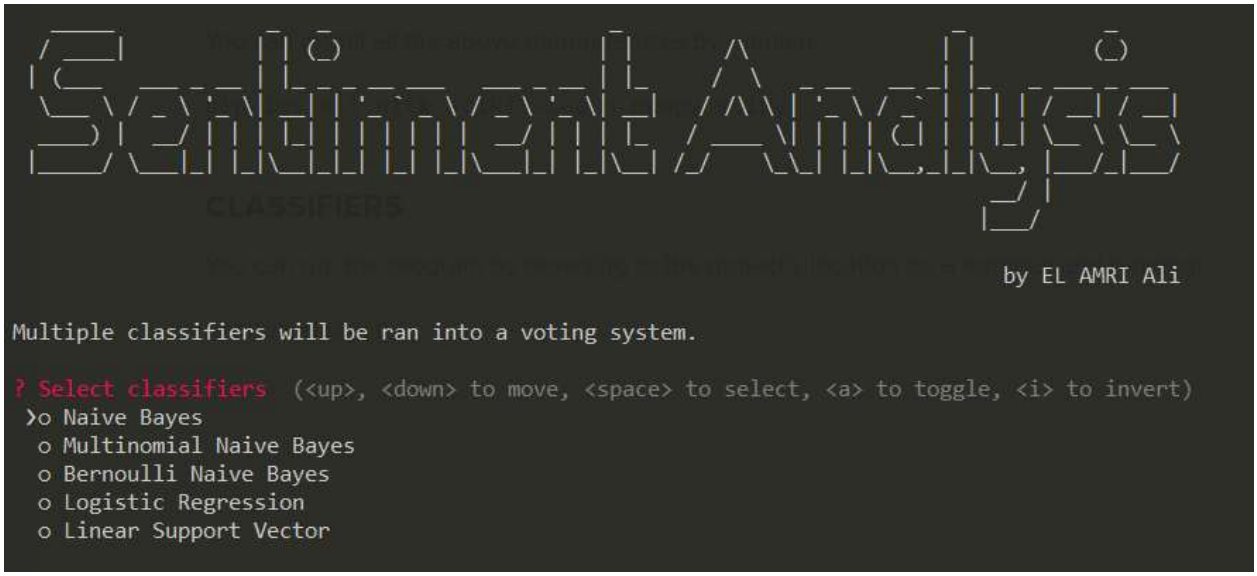
```
pip install nltk scikit-learn numpy scipy
```

CLASSIFIERS

You can run the program by browsing to the project's location on a terminal and running:

```
python Main.py
```

Which will output the starting screen:



And select which classifiers you which to make use of by moving up and down and pressing spacebar to toggle on or off.

Using more than one classifier was a challenge of mine to get familiar with the different approaches and tuning of each one in order to increase accuracy but why limit myself to running a single classifier at once, let's get crazy and use them all at once!

This approach raised the issue of how to pick which classifiers' results to chose i.e. the three naive bayes classifiers return that a comment is positive while the other two state that it is neutral or negative. The fact that the number of chosen classifiers is odd was a decision of mine to make sure than any two decisions or results (positive, neutral, negative) could occur equally, one will always trump the other two.

A sixth classifier was created called VoteClassifier whose main task is to determine which result is agreed upon between all the previous classifiers and even set a confidence level for each.

```
def classify(self, features):
    votes = []
    for classifier in self._classifiers:
        vote = classifier.classify(features)
        votes.append(vote)

    grouped_votes = [[k, ] * v for k, v in Counter(votes).most_common()]
    sentiments = []
    for vote in grouped_votes:
        sentiments.append({
            'sentiment': vote[0],
            'confidence': len(vote) / len(votes)
        })

    return sentiments
```

It will append to each comment a list of the different sentiments (positive, neutral, negative) and the confidence level which is basically the percentage of the classifiers that produced that sentiment.

E.g. 2 classifiers resulted in a positive sentiment, another one outputted neutral while the remaining two stated it was negative, the comment is then 40% positive, 40% negative and 20% neutral. But as I said before, this case of equality between two sentiments was virtually impossible to get (trust me, I did the manual testing myself)

Let's get back to the terminal screen where once you toggle on the classifiers to be used, you are prompted to select which corpus to work on.

CORPORA

Another challenge when it comes to working with classifiers is training and testing. Although it may seem biased, the selection of the data set to train and test has to be as closed to the real world application as possible, in our case it was about youtube comments which are, as we all know, are a complex mixture of opinionated reviews and very often a language only deciphered by its writer.

The choice of a corpus made by pressing either 1 or 2 button on the numpad and the selection is reflected and the answer in the last line.

Although I never mixed the two and only used one at a time to see how the results differ between the two, which was very interesting, each one had a specific purpose:

IMDb Reviews:

A hand selected dataset that contains reviews along with their associated binary sentiment polarity labels, meaning only positive and negative (sadly) which is intended as a benchmark for sentiment classification. This constitutes the “opinionated review” part that we talked about earlier, we’ll discuss the slang-ish and informal aspect later on.

This data set is huge, it contains 50.000 pre classified movie reviews divided equally between training and testing and then between positive and negative reviews (meaning 12.500 for each subset: 12.500 positive for training, etc...)

It was obviously too large to work on and took about a full day to train on my last computer (may it rest in pieces, get it ? :p) but I have added an extra layer of configuration to tune the training and testing sets on the fly located in Config.py

```
class Config:

    # Number of reviews to test upon, 25.000 being the limit
    # this value will be divided between positive and negative data
    # i.e. training_set_size = 8000 means 4000 positive reviews and 4000 negative ones
    training_set_size = 25000

    # the same applies of the testing_set_size
    testing_set_size = 3000
```

We will talk a bit more about the process of training and testing and all the underlying processes tokenization, extracting features, etc... later on.

Twitter US Airlines:

The choice of this data set was to see how it would handle the internet-fast-talk and informal aspect and use of slang is youtube comments, this data set contains about 14.600 entries on which we apply the same limitation rules of the previous data set due to its size and the varianting performances of the different computers on which this program will run.

Once the choice of the corpus is made, pressing enter will show the following screen:

```
 _____                _   _                 _                 _                 _
/  ___|              | | (_)               | |               | |               (_)
\ `--.  ___ _ __ | |_ _ _ __ ___   ___ _ __ | |_    / \   _ __   __ _| |_   _ ___ _ ___
 `--. \/ _ \ '_ \| __| | '_ ` _ \ / _ \ '_ \| __|  / /\ \ | '_ \ / _` | | | | / __| / __|
/\__/ /  __/ | | | |_| | | | | | |  __/ | | | |_  / ____ \| | | | (_| | | |_| \__ \ \__ \
\____/ \___|_| |_|\__|_|_| |_| |_|\___|_| |_|\__|/_/    \_\_| |_|\__,_|_|\__, |___/_|___/
                                                                           __/ |
                                                                          |___/

                                                                by EL AMRI Ali

Corpus:         Twitter

Classifiers:    Naive Bayes
                Multinomial Naive Bayes
                Logistic Regression
                Linear Support Vector

? Which action do you wish to perform
  1) Training: Train the classifiers using the corpus.
  2) Testing: Assess the classifiers' accuracy using the corpus.
  3) Analyzing: Analyze the Sentube data set using the classifiers and the corpus
  Answer: 1
```

Note the user is constantly reminded of his previous choice for consistency purposes, although it did not implement a going back option as I just thought of it while writing this report, pressing Ctrl+C will cancel the current program and the user will have to rerun it in order to select other options.

TRAINING

Let's dive in into some details! Training is probably the most crucial part of working with classifiers and data sets and sentiment analysis in general, as it requires some pre-processing and other tasks before feeding the data to the different classifiers.