TEXT MINING PROJECT: SENTIMENT ANALYSIS IN TWEETS AND EXTERNAL LINKS

Amit Patil, Email: f5amit@gmail.com
Aswin Pai, Email: aswinpai@gmail.com
Julia Semenova, Email: juliasemenova13@gmail.com
October 8, 2016

1 Introduction

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 over 200 million registered users - out of which 100 million are active users and half of them log on twitter on a daily basis - generating nearly 250 million tweets per day. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analysing the sentiments expressed in the tweets. Analysing 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.
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 unlabelled data samples (tweets) according to whichever pattern model best describes them.
The features that can be used for modelling 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, word lengthening, question marks, presence of url in tweets, exclamation marks, internet emoticons and internet shorthand/slangs.
There are several metrics proposed for computing and comparing the results of our experiments. Some of the most popular metrics include: Precision, Recall, Accuracy, F1-measure, True rate and False alarm rate (each of these metrics is calculated individually for each class and then averaged for the overall classifier performance.) A typical confusion table for our problem is given below along with illustration of how to compute our required metric.

2 Targets

The main objectives of this project are to:

1. To perform hyperlink resolution on external links that resolve to text sources e.g., journalist web page/blog, on a dataset consisting of 2000 microposts about a selected keyword by the user and,

2. further determine the sentiment of each individual micropost.

Once this operation is done, the overall sentiment of the dataset should be represented as a visualisation to the end-user.
In this project, we want to implement techniques to determine sentiment in microposts, which will be tweets. One method will one using a Naive Bayes classifier and another will be using a wordlist(AFINN). Detailed explanations are given below.

2.1 Naive Bayes Classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y|x_1,x_2,...x_n) = \frac{P(y)P(x_1,...x_n)|y)}{P(x_1,...x_n)}$$

Using the naive independence assumption that,

$$P(x_i|y,x_1,x_2,...x_{i-1},x_{i+1},...x_n) = P(x_i|y)$$

,
for all i, this relationship is simplified to,

$$P(y|x_1,x_2,...x_n) = \frac{P(y)\prod P(x_i|y)}{P(x_1,...x_n)}$$

Since $P(x_1,...x_n)$ is constant given the input, we can use the following classification rule:

$$P(y|x_1,x_2,...x_n) \propto P(y)\prod P(x_i|y)$$

The resulting class of the entire vector is calculated by product:

$$P(y|x_1,x_2,...x_n) = P(y|x_1) \times P(y|x_2) \times ... \times P(y|x_n)$$

2.2 AFINN

Sentiment analysis of microblogs such as has recently gained a fair amount of attention. One of the simplest sentiment analysis approaches compares the words of a posting against a labeled word list, where each word has been scored for valence, – a "sentiment lexicon" or "affective word lists". There exist several affective word lists, e.g., ANEW (Affective Norms for English Words) developed before the advent of microblogging and sentiment analysis.
AFINN is a Python module to add sentiment analysis using a wordlist. A detailed explanation can be found at [1].

Snippet from the wordlist, where a set of words have a pre-defined polarity:
astounding 3
astoundingly 3
astounds 3
attack -1
attacked -1

2.3 Software used:

Python programming language was with no specific IDE. The program was run via command line in Linux or windows. JSON files are used to collect and store data. For testing purposes, Jupyter notebook was used.

3 Design

3.1 Structure

The program functions in the following way: On the web interface, we enter a keyword based on which we want to perform sentiment analysis. The program then collects tweets related to the keyword in a limited timeframe. The information from the external links in a tweet is also extracted. All the information is then stored in a JSON file. Using this information, a classifier determines the overall sentiment related to the keyword. As an output, the user receives a pie chart of degrees of sentiment and a wordcloud.

3.2 Packages Used

There are several packages used in the project. They are the following:

1. TextBlob(Used in classifier approach): TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

2. AFINN(Used in second approach): AFINN is actually a Python module, which contains a wordlist with sentiment scores between 5 and -5. These scores are used to compute the overall sentiment in mocroblogs.

3. NumPy: is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

4. URLLib: is a package that collects several modules for working with URLs.

5. Pandas: is a Python package providing fast, flexible, and expressive data structures designed to make working with âĂIJrelationalâĂİ or âĂIJlabeledâĂİ data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

6. NLTK library: suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English.

7. BeautifulSoup: is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.

8. MatPlotLib: is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

9. WordCloud: is a python package that produces a figure containing the most frequently used words.

3.3 JSON Files Used

There are no pre-defined JSON files, they are dynamically created and used during the run of the program to collect and store tweets and information relating to the keyword used.

3.4 Python Files Used

There are four python files used:

1. sentiment.py: The code that determines the sentiment of the collected data from the tweets.
2. url_sentiment.py: The code that determines the sentiment of the external links, if present, if tweets.
3. mainfile.py: The mainfile where the keyword based on which tweets are retrieved is used.
4. twitterauth.py: Authentication to retrieve tweets from twitter and also sets the limit on the number of tweets to be collected.
5. hashtag.py: Mines the hashtags used in tweets, counts them and displays the data in the form of a bar graph.
6. create_gui.py: Once all the data has been collected, this file is used to create an impromptu gui to show the results.
7. get_url.py: Contains code which extracts url links in a tweet. The link is opened and relevant information is retrieved using certain packages.
8. classify.py(Used in the NB classifier approach): Contains the classifier which is used to classify the text of tweet and the external link.

4 Implementation

Implementation of the project revolved around the method by which the text would be classified. Initially, a Naive Bayes Classifier was used that computes the probability of the sentiment in the tweet and the external link. The classifier is first trained on a train set and then implemented on a train set. Initially, an accuracy of 80% was achieved. However, when the test set was expanded to include higher range of sentences containing varying range of sentiments, the accuracy of the classifier dropped. To ensure a higher range of words and sentiments are incorporated, *TextBlob* package was used to accomplish tasks such as POS tagging and determining the polarity of text.
With further experimentation, it was discovered that better results were achieved when the *AFINN* package was used rather than TextBlob. AFINN consists of wordlist with a sentiment score between 5 and -5, where 5 is the most positive and -5 is the most negative. With the use of AFINN, the sentiment classification seemed to be more precise, ostensibly due to the pre-defined scores. This was observed by comparing the pie charts of the classification: In case of the classifier, tweets were largely classified as "Neutral", while

with the use of AFINN, the so-called "Neutral" tweets from under the classifier were being correctly classified as "Positive" or "Negative". It then followed with the use of AFINN package that a classifier itself was not required.
The running process of the program is the following: When a tweet is received, the text of tweet is extracted from the tweet itself, hashtags(which is a main criteria for collecting tweets) and external links. *Beautiful Soup* package is used to strip the webpage and retrieve the important information. Sentiment analysis is then performed on the entire text.
The text will eventually be classified as Positive or Negative or Neutral. As an output, the following is received:

- A Wordcloud containing the most used words in tweets,
- A bar graph depicting the frequency of hashtags,
- A text file containing links to each individual tweet and their corresponding sentiment,
- A pie chart depicting percentage of sentiment of the total set of tweets

A confusion matrix depicting the accuracy of the classifier from the first approach is also given below.

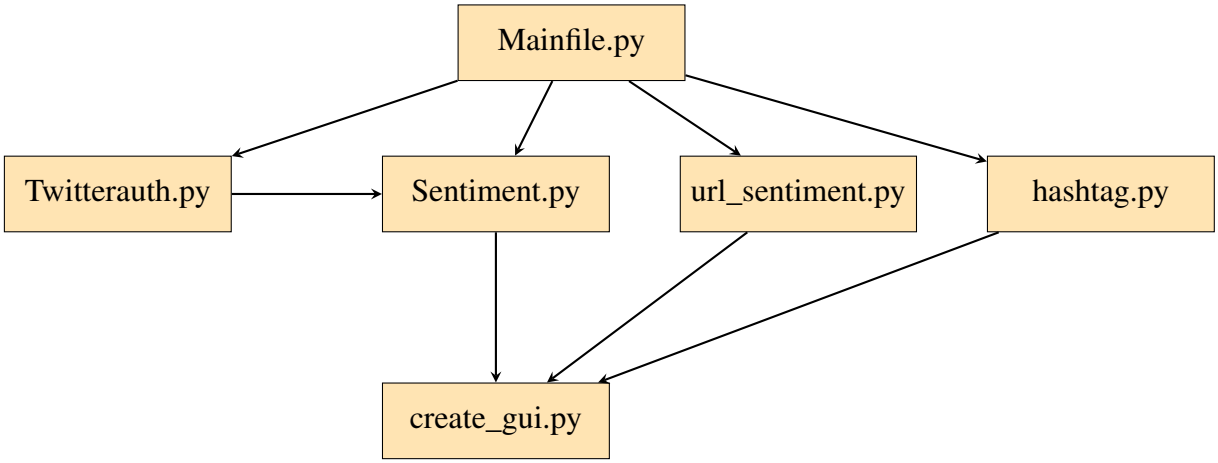

Figure 1: Brexit sentiment analysis

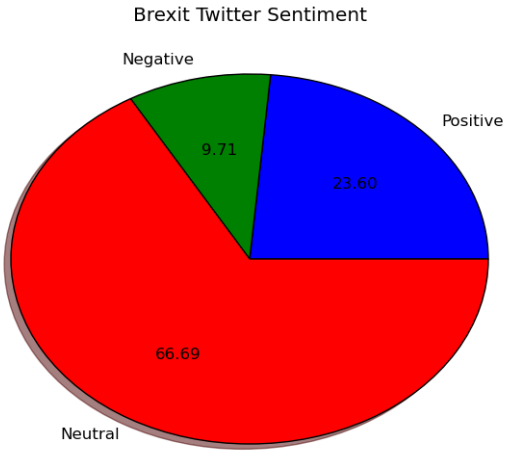

Figure 2: Donald Trump Sentiment Analysis

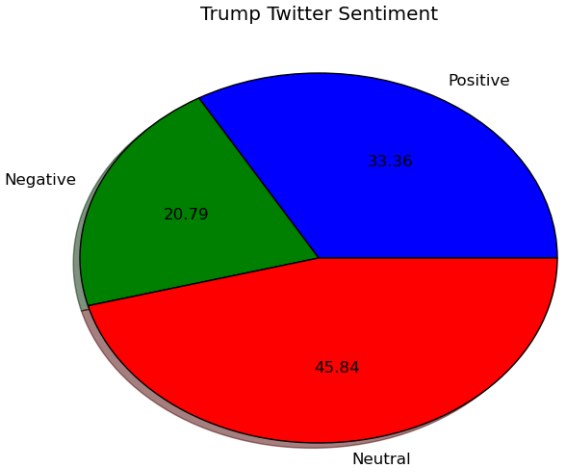

Figure 3: Brexit WordCloud

Figure 4: Donald Trump WordCloud

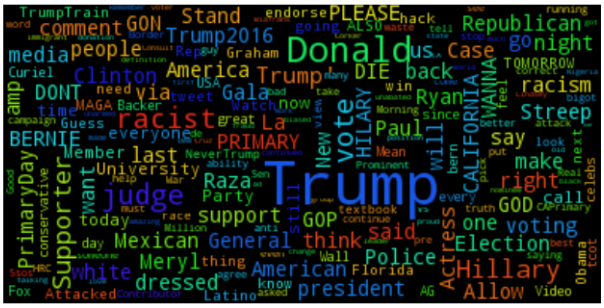

Figure 5: Confusion Matrix

N = 22	Predicted NO	Predicted YES	
Actual NO	TN = 5	FP = 3	8
Actual YES	FN = 6	TP = 8	14
	11	11	

Figure 6: HashTag Bar Graph

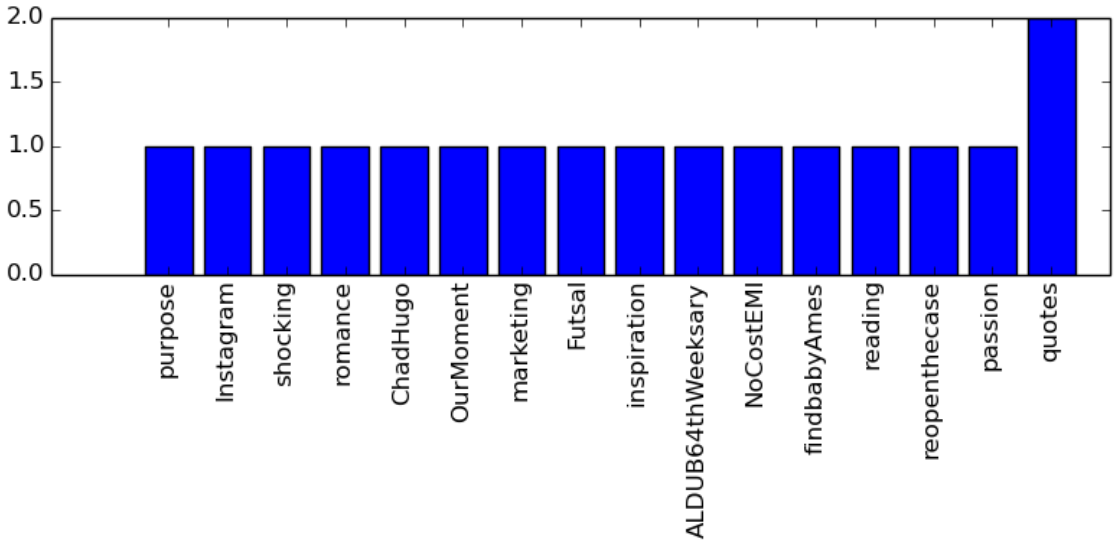

Figure 7: Sentiment Analysis using AFINN

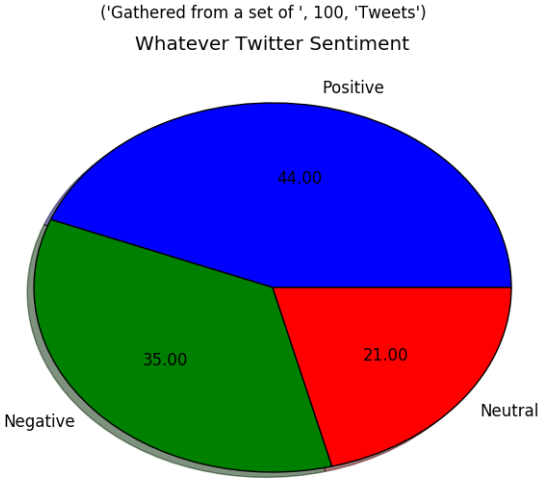

Figure 8: Improved WordCloud

5 Conclusion

The project was designed and implemented. The primary goals were achieved and a few other implementations were added. The classifier was successful in achieving modest results. The implementation using AFINN resulted in more accurate classification of sentiment.

References

[1] Finn Årup Nielsen, *A new ANEW: evaluation of a word list for sentiment analysis in microblogs*, booktitle = Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages, pages = 93–98, year = 2011, editor = Matthew Rowe and Milan Stankovic and Aba-Sah Dadzie and Mariann Hardey, volume = 718, series = CEUR Workshop Proceedings, month = May,

[2] Official NLTK documentation.