FINAL REPORT

SENTIMENT ANALYSIS VIA SOCIAL MEDIA

TEAM MEMBERS:

RITIKA JAIN

SAYALI THORAT

NEHA SINGH

KUMAR KRISHNAMURTHY

**Introduction:**

Sentiment analysis of in the domain of micro-blogging is a relatively new research topic so there is still a lot of scope for further research in this area. Large amount of related prior work has been done on sentiment analysis of user reviews, web blogs, articles and general phrase level sentiment analysis. These totally differ from twitter mainly because of the limit of 140 characters/tweet which forces the user to express opinion compressed in very short text. It uses supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. There is some work has been done on unsupervised and semi-supervised approaches, and there is a lot of scope of improvement. Many researchers testing new features and classification techniques often just compare their results to base-line performance. We need proper and formal comparisons between these results arrived through different features and classification techniques to select the best features and most efficient classification techniques for particular applications.

Sentiment Analysis is the process of predicting whether a piece of information where the text, most commonly indicates a positive, negative or neutral sentiment on the topic. In this project, we will go through making a Python program that analyses the sentiment of tweets on a particular topic. User will be able to input a keyword and get the sentiment on it based on the latest tweets that contain the input keyword.

**Problem Definition:**

In this project we will focus on the common binary problem of identifying the positive, negative and neutral sentiment that is expressed by a given text toward a specific topic. The texts that we deal with in this project, it must express either positive, negative or neutral sentiment. There are other Tasks that allow a text to be neutral about a specific topic, or even totally objective which express no interest in the topic at all. So we narrowed it down the problem like this, we get a classic binary classification case. In this, we will use manually entered text: i.e. twitter posts. We used collection of Twitter short messages, on pericular topic and sentiment that is expressed toward it. We used three classifiers:

Naive Bayes and SVM over a set of features that were extracted from the texts using techniques taken from the field of natural language processing (NLP).

|  |  |  |
| --- | --- | --- |
| **Tweet** | **Topic** | **Polarity** |
| Wow! I love his acting and his choice of movies!! Just too good to be true… | Tom Hanks | Positive |

**Table 1: Example of Tweet**

**Key contributions:**

**Basic programming knowledge**

As we know, Python is highly involved in this mini-project, we don’t to have a deep knowledge in the language, as long as you have basic programming knowledge.

**Installed tools**

For this program, we will need Python to be installed on the computer. We will be using the libraries tweepy, nltk, re, csv, time. You need to install the first two libraries. Others already come with the Python interpreter.

**Data set splitting concept**

We need to know [the difference between Training and Test data sets](https://medium.com/datadriveninvestor/what-are-training-validation-and-test-data-sets-in-machine-learning-d1dd1ab09bae) and in what context each one is used.

**Machine learning**

We have used Naïve Bayes and SVM model for analysing the sentiments of tweets.

**Naive Bayes:**

In machine learning, Naive Bayes classifiers use simple probabilistic classifiers based on applying Bayes theorem with strong independence assumptions between the features. Naive Bayes classifiers require a number of parameters linear in the number of variables in a learning problem. The basic idea of Naive Bayes technique is to find the probabilities of classes assigned to texts by using the joint probabilities of words and classes.

We have used the Multinomial Model which is typically used for discrete counts. In text classification, we extend the Bernoulli model further by counting the number of times a word appears and the number of words rather than saying 0 or 1 if word occurs or not.

**SVM:**

Support Vector Machine is a supervised machine learning algorithm that can be used for both classifications, regression challenges. Classification is predicting a label/group and Regression is predicting a continuous value. SVM performs classification by finding the hyper-plane that differentiates the classes we plotted in n-dimensional space.

We are going to use cross validation and grid search to find good hyper parameters for our SVM model. We need to build a pipeline to don’t get features from the validation folds when building each training model.

**Datasets**

In this project our intention is to focus on the polarity of the problem (positive/negative/ neutral). Each entry contains the tweet, the sentiment polarity (positive / negative/Neutral) and the topic subject of the sentiment polarity. Table 1 shows an example such an entry.

We are using the real-time dataset which gives more flexibility and our algorithm train the data accordingly.

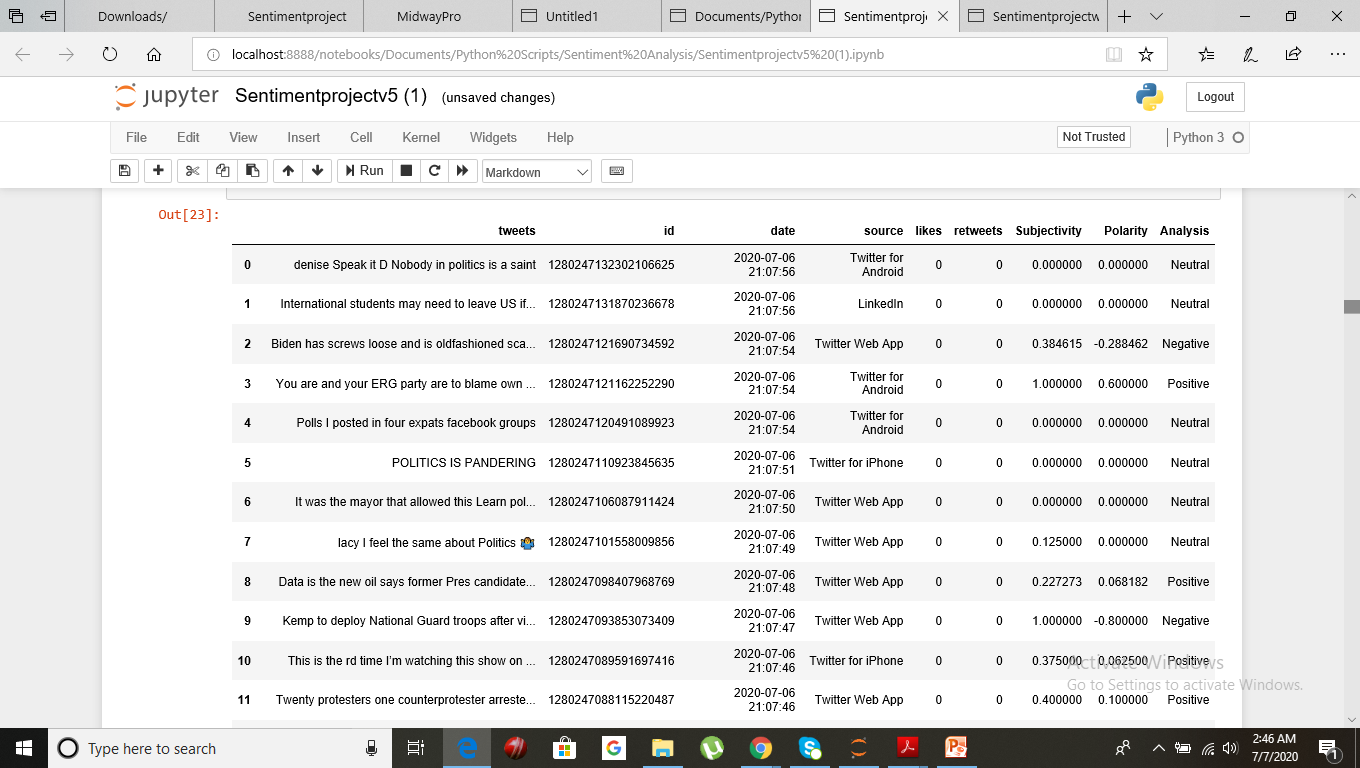
Here due to legal constraints, the text of every tweet was not in the original corpus, but only the tweet ID. To get the text of tweet, we should use the Twitter API accordingly. We have already discarded the retweets as they have few repetitions in retweets, this way we have handled the duplicity. The main challenge of processing tweets emerge because of the size restriction is only for 140 characters. This restriction Makes people use shortcuts, omit prepositions and use emojis. In Twitter, People are often use hashtags attached to a single word expression, for referring to a trending topic.

**Data set pre-processing**

Now that we have the corpus of tweets and all the resources that could be useful, we can pre-process the tweets. It is a very important since all the modifications that we are going to during this process will directly impact the classifier’s performance. The pre-processing includes cleaning, normalization, transformation, feature extraction and selection, etc. The result of pre-processing will be consistent and uniform data that are workable to maximize the classifier's performance. All of the tweets are pre-processed by passing through the following steps in the same order:

* Remove Retweets
* Allow only English tweets
* Remove duplicate tweets
* Remove tags
* Remove hashtags and hyperlinks

**Fig: Represents the dataset after Analysis**



**Finding the best machine learning model for the problem objective**

As we have proceeded till this part in our project and understanding our data from twitter, the next thing we did is identified the algorithms that are applicable and practice to apply in our project. Some of the elements we used are:

* The accuracy of model
* The interpretability of the model
* The complexity of model
* How long does it take to build, train and test the model
* How long does it take to make predictions using the model

**Implemented machine learning algorithms:**

We have set up a machine learning models that compares performance of each algorithm on the datataset using a set of twitter dataset. The best solution we arrived at is to have a service running that does this in intervals when new dataset is added.

**Support Vector Machines**

SVM is a supervised machine learning model algorithm for two-group classification problems. We used SVM as set of labelled traning data for each category, and then categorised them. Here we categorised vectors in multidimensional space. We transformed a piece of text into a vector of numbers so we can run SVM with them.

We’ve done that, every text in our dataset is represented as a vector with thousands (or tens of thousands) of dimensions, every word representing the frequency of one of the words of the text. This is what we put into our SVM for training. We can improve this by using pre-processing techniques, like stemming, removing stopwords, and using n-grams.

The feature vectors are arranged, then we used kernel function for our vector.  In our project, our data was arranged in concentric circles, so we chose a kernel that matched all those data points

Now we have taken our set of labelled texts, convert them to vectors using word frequencies, and put them to the algorithm, which used our chosen kernel function and so it produces a model. Then, when we had a new unlabelled text that we want to classify, we converted it into a vector and give it to the model, which will output the tag of the text.

**Naive Bayes**

We have also used Naive Bayes as it is a probabilistic method. Here we were able to perform a high number of probabilistic based calculation in a very short period of time as its training time is very fast, so to come up with the prediction it is faster than the SVM.

In this model, we have used these techniques as it allows Naive Bayes to perform at the same level. It can remove stopwords, lemmatizing words, using n grmas.

We have assumed that every word in a sentence is independent of the other ones. Here we have no longer looking at entire sentences, but rather at individual words.

We have observed, all of these individual words show up several times in our training data, and we have calculated them.

We have also used Naive Bayes as it is a probabilistic method. Here We were able to perform a high number of probabilistic based calculation in a very short period of time as its training time is very fast, so to come up with the prediction it is faster than the SVM. So the conclusion here was that it is very efficient as it takes less time in the training and testing phase. It can deal with data that are in very high dimensions. But we concluded that the accuracy was better in case of SVM model whether we take 5000 row data set or 500 row data set.

“Naive bayes accuracy was 58 % while the SVM model accuracy is 70% only.”

SVM model accuracy was higher than the naive bayes accuracy.

**Training and testing the model**

In our model, dataset meets the preceding two conditions, our goal was to create a model that generalizes well to our new data. Our test set serves as a proxy for new data.

As in our model, task of Sentiment Analysis is one that works on textual data, here we did lot of text processing. In fact, both our Test and Training data in our model will merely comprise of text.

We chose to start our model with the Test set in order to get all warmed up for the Training set extraction part, as it relies more on the API. Basically, in this it is a function that takes a search keyword (i.e. string) as an input, searches for tweets that include this keyword and returns them as twitter data. Status objects that we can iterate throughout the model.

But the condition here, was that Twitter limits the number of requests you can make through the API for security purposes. This limit is 180 requests per 15-minute window.

This means, we can only get up to 180 tweets using our search function every 15 minutes, which should not be a problem, as our Training set is not going to be that large anyway. For the sake of simplicity, we can take the whatever number of tweets we want to take for now, not exceeding the allowed number of requests.

As expected, the function we used has returned a list of tweets that contain our search keyword. This print out five tweets that contain our search keyword on the Terminal of our IDE.

**Preparing The Training Set**

We have used our Twitter API instance. We used downloadable Training set. The tweets that we downloaded were all labelled as positive or negative, depending on the content. This was exactly what the Training set was set for.

A Training set is very important for the success of the model. Here Data is needs to be labelled properly with no inconsistencies or incompleteness or no error, as training will rely mostly on the accuracy of the data and the manner of acquisition.

For this task, we can allow user to enter the most trending tweets that are used worldwide that is 5000 hand-classified tweets, which makes it quite well founded. Here, the challenge we faced was Twitter does not allow storing tweets on a personal device. Corpus includes a keyword (topic of the tweet), a label (pos/neg) and a tweet ID number for every tweet (i.e. row in our CSV corpus).

Let’s finish up our work by running the classifier on the n number of tweets that we downloaded from Twitter, according to our search term we used, and getting the majority vote of the labels returned by the classifier. After this outputting the total positive or negative percentage (i.e. score) of the tweets we have collected.

**Model Evaluation:**

As part of our project, we evaluated multiple ML Algorithm models for the purpose of Sentiment Analysis on Dynamic Tweets such as:

* Supervised
* SVM
* Naive Bayes
* Unsupervised
* Keras

Since, we were working on dynamic live twitter datasets. Following is the observation we noted related to different Algorithms:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **SVM** | **Naive Bayes** | **Keras** |
| Training Time | High | Fast | Depends on EPOCHS |
| Accuracy (5000 row data set) | 70% | 58% | 43% |
| Accuracy (500 row data set) | 59% | 55% | 24% |
| Observation | Accuracy always high compared to other 2 Algorithms.  Higher the Dataset volume. Higher the Accuracy.  Even when dataset had no region or subject guardrails | Accuracy always lesser compared to SVM.  Randomly in some cases for smaller dataset scenarios, Naïve Bayes performed better than SVM. | Our Model training accuracy was always not improving through the epochs for live dataset from twitter as the dataset had no region or subject guardrails.  When our Model was trained with dataset which had specific set of words related to region or subject. It trained well and the accuracy was improving and was a better performing Model. |

**Analysis of results:**

The field of sentiment analysis is new research direction due to large number of real-world applications where discovering people’s opinion is important in better decision-making. The development of techniques for the document-level sentiment analysis is one of the significant components of this area. Recently, people have started expressing their opinions on the Web that increased the need of analysing the opinionated online content for various real-world applications. A lot of research is present in literature for detecting sentiment from the text. Still, there is a huge scope of improvement of these existing sentiment analysis models. Existing sentiment analysis models can be improved further with more semantic and common-sense knowledge.

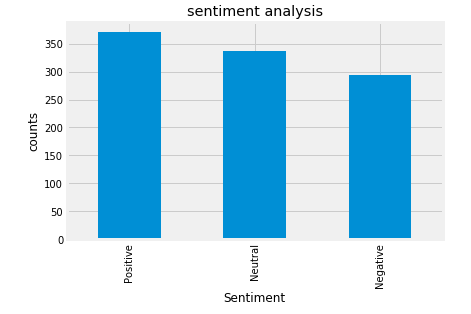
We completed by running the classifier (i.e. NBayes Classifier) on the 100 tweets that we downloaded from Twitter, according to our search term, and getting the majority vote of the labels returned by the classifier, then outputting the total positive or negative percentage (i.e. score) of the tweets.

We have created a Twitter Sentiment Analysis Python program. We have touched on the accuracy (i.e. evaluate the model) using SVM and Naïve Bayes model. Sentiment Analysis is an interesting way to think about the applicability of Natural Language Processing in making automated conclusions about text. It is being utilized in social media trend analysis and, sometimes, for marketing purposes. Making a Sentiment Analysis program in Python is not a difficult task, thanks to modern-day, ready-for-use libraries. Our project is a simple explanation to how this kind of application works.

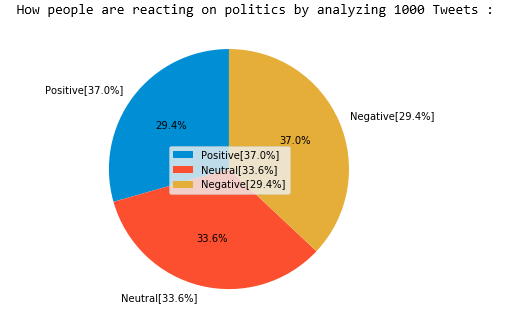
**RESULTS:**

**For considering the Keyword as Politics and number of tweets as 5000.**

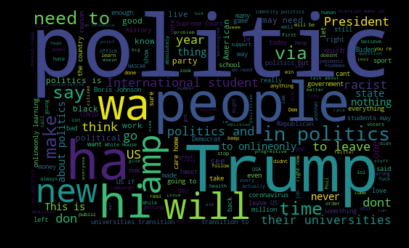
1. **Bar Graph**

****

1. **Pie Chart**

****

1. **Word Cloud**

****