# Semantic Linking for Event-Based Classification of Tweets

Anas Siddiqi Muhammad Zohaib Khan Umair Javed

siddiq01@gw.uni-passau.de
khan06@gw.uni-passau.de
javed01@gw.uni-passau.de

## 1 Abstract

Social networking service or Social media is a platform that enables its users to share information, ideas and other form of expressions via virtual networks and communities. Social media generate huge amount of textual information every day and mining on social media streams is a challenging task as compared to traditional text streams, especially the task of detecting an event within a data stream from twitter. In this paper we are going to detect events within twitter stream using machine learning approach. Our main approach will be to use semantic linking from different ontologies like DBpedia to named entities in tweets aiming to enrich textual information within a tweet, semantic linking of twitter posts also promises benefits for other applications on the social web and machine learning application. We will use different supervised machine learning techniques to detect events and analyze the effect of semantic linking with named entities in tweets.

## 2 Introduction

Social media has gained a lot of attention today. The popularity of social media is increasing at fast pace and people can access and use social media anywhere and anytime. Twitter is a social media platform on which millions of users share their views, ideas, and content. In order to monitor trending topics and in which way events are affecting people's life, automated topic and event detection is an emerging field of research. Therefore, event detection is an important task and there are a lot of new ways proposed to handle this task in literature.

In this article, we provide work for linking tweets to related events and analyze them to contextualize events. Next, we propose strategies that evaluate the semantics of tweets and related events to semantically make individual twitter activities meaningful. A large-scale semantics evaluation with lot of different machine learning methods confirms the benefits of our approach and shows that our way of addressing the issue of relating tweets with high precision and reach to related event.

Event detection in twitter is a difficult task due to the lack of contextual information in tweets and most of the tweets are not related to events. In addition, traditional text mining techniques are not suitable, because of the short length of tweets, the large number of spelling and grammatical errors, and the frequent use of informal and mixed language.[7]

We chose supervised machine learning techniques to handle this problem and we will focus on multi-class classification, or Ordinal classification of tweets data as each event type has its own class. The goal is to learn and use machine learning algorithms such as Naïve Bayes (NB), Support Vector Machine (SVM) and Artificial Neural Networks (ANN) and train these algorithms using tweets dataset, then validate results via Confusion Matrix. Afterwards replace the named entities in data with their semantic types from different ontologies of DBpedia and calculate the impact of semantic linking of named entities to the accuracy of classifiers.

# 3 Semantic Linking of Named Entities

For semantic analysis there are number of extractors available that can help you to analyze your data semantically.[3] There is the long list that we cannot show here but the most popular shown below:

- AlchemyAPI [1]
- Dandelion API [2]
- Lupedia [3]
- OpenCalais [4]
- Saplo [5]
- dbpedia [6]
- TextRazor [7]
- Wikimeta [8]
- Yahoo! Content Analysis [9]
- Zemanta [10]

Popular ontologies are N.E.R.D (named entity Recognition and Disambiguation), IBM Watson and YAGO. The NERD ontology consists of two building blocks: the NERD core and the derived NERD axioms. The list of NERD core classes is defined as:
Thing, Amount, Animal, Event, Function, Location, Organization, Person, Product and Time classes. NERD also provides the framework that standardizes the output of 10 different NLP (Natural Language processing) extractors, which are publicly available on the Internet.[5] Provide us the wide range of options to choose the among the extractors.[6] Also, options to combining output of all the given extractors and give one entity per document, which filters out the duplicates entities. Our approach was mainly based on the development with the NERD ontology, which provides a common interface for annotation elements, by the mean of REST API. That could be used to access the single output of these extractors altogether. The NERD application is available online at [11] also with the NERD API libraries for different platforms. While working on NERD also there were few problems we faced. Some of it's extractor were in active or just not working or occupied by some other companies, Like the AlchemyAPI were acquired by IBM Watson, Saplo was acquired by Strossle. But still many of its extractor that are using the ontology of N.E.R.D are still in working form. NERD's response looks like

```
{
        "idEntity": 32735201,
        "label": "Munich",
        "extractorType": "Place,PopulatedPlace,/location/location",
        "nerdType": "http://nerd.eurecom.fr/ontology#Thing",
        "uri": "http://de.wikipedia.org/wiki/München"
        "extractor": "textrazor",
    }
```

IBM Watson is one of the most popular and biggest application of machine learning we took some help for the verification of our work by comparing or entities result and there ontologies. Which was really helpful for the cross validation of our results. Watson gives you a verity of option to play with. We use there discovery API, which provide you the deep sentiment and semantic linking of our data.

[1] http://www.alchemyapi.com
[2] https://dandelion.eu/
[3] http://lupedia.ontotext.com/
[4] http://www.opencalais.com
[5] http://www.saplo.com/
[6] http://dbpedia.org/spotlight
[7] https://www.textrazor.com/
[8] http://www.wikimeta.com
[9] http://developer.yahoo.com/search/content/V2/contentAnalysis.html
[10] http://www.zemanta.com
[11] http://nerd.eurecom.fr

# 4 Event Based and General Twitter Datasets

We downloaded the twitter datasets from social sensor's website[12] [2]. We downloaded tweet ids for three different social events from 2012, including US Elections, Super Tuesday and Fifa Cup.

We downloaded the event related tweets data from twitter by tweet-ids we got from social sensor website using *tweepy* library[13]. We also obtained a general tweets dataset from twitter stream[14]. Following is the number of tweets that are included in each event based dataset.

| Event | Tweets |
|---|---|
| FiFa Cup | 200k |
| Super Tuesday | 400k |
| US Elections | 3200k |
| General Category | 300k |

Table 1: Twitter Dataset

# 5 Detecting and Classifying Event Tweets

This section describes the used approach to detect tweets which are related to events. Given a dataset of tweets, the main steps include:

- Preprocessing
- Named entity linking and replacement
- Feature extraction
- Tweet classification
- Evaluation

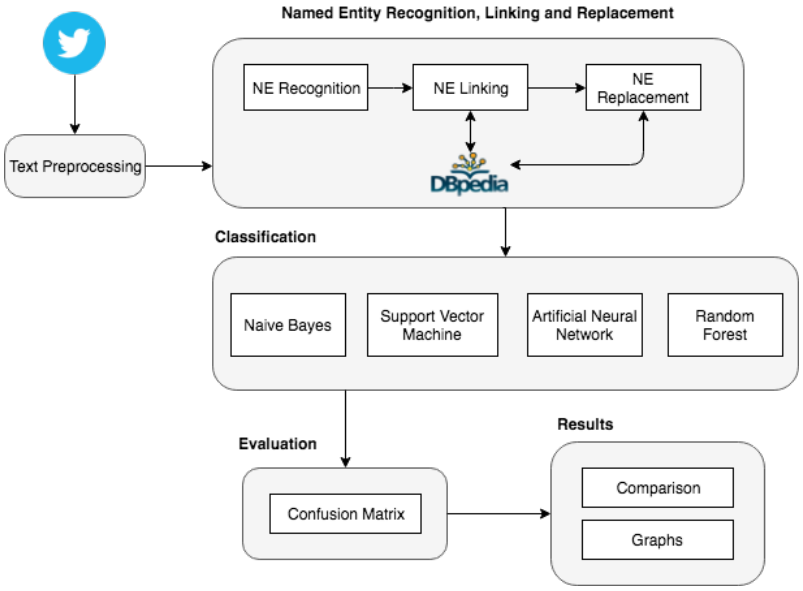


Figure 1: Architecture

[12] http://www.socialsensor.eu/results/datasets
[13] http://docs.tweepy.org/en/v3.6.0/
[14] https://archive.org/details/twitterstream

## 5.1 Preprocessing

In order to extract ontologies for named entities of our tweets data, we cleaned the tweets text of our datasets. First we removed information that is specific to twitter, like user mentions, hashtags, emoticons and URIs, then removed any extra white spaces in the data, and converted to lowercase. We removed any characters that appeared more than two times in a word, and finally using lemmatization and stemming to bring all the words to simple grammatical form.

## 5.2 Named Entity Linking and Replacement

After the preprocessing step, we called NERD APIs to recognize Named Entities and link them to DBpedia for entity linking. We used three APIs of NERD in order to get the ontologies for named entities in tweets data:

### 5.2.1 Document API

In first step we call the document API[15] of NERD , we send data in text form to this API and it returns a document id. Data size limit that we used to send to this API is 170 KB, which we found to be the threshold for getting highest possible successful responses from NERD with minimum number of calls.

### 5.2.2 Annotation API

In second step we call the annotation API[16] of NERD, we send the document id we got from first API (5.2.1) to this API, and it returns an annotation id. NERD uses different formats of ontologies depending upon the extractor type. We used *textrazor* as the extractor type to get the ontologies of named entities from NERD.

### 5.2.3 Entity API

In third step we call the entity API[17] of NERD, we send the annotation id we got from second API (5.2.2) to this API, and it returns a *json* response containing semantic links to different classes(generic and specific) for all the named entities that it recognizes from the data we sent in API (5.2.1).

Below is a data flow and diagram that provided an overview of Named Entities Extraction and Linking in the data.

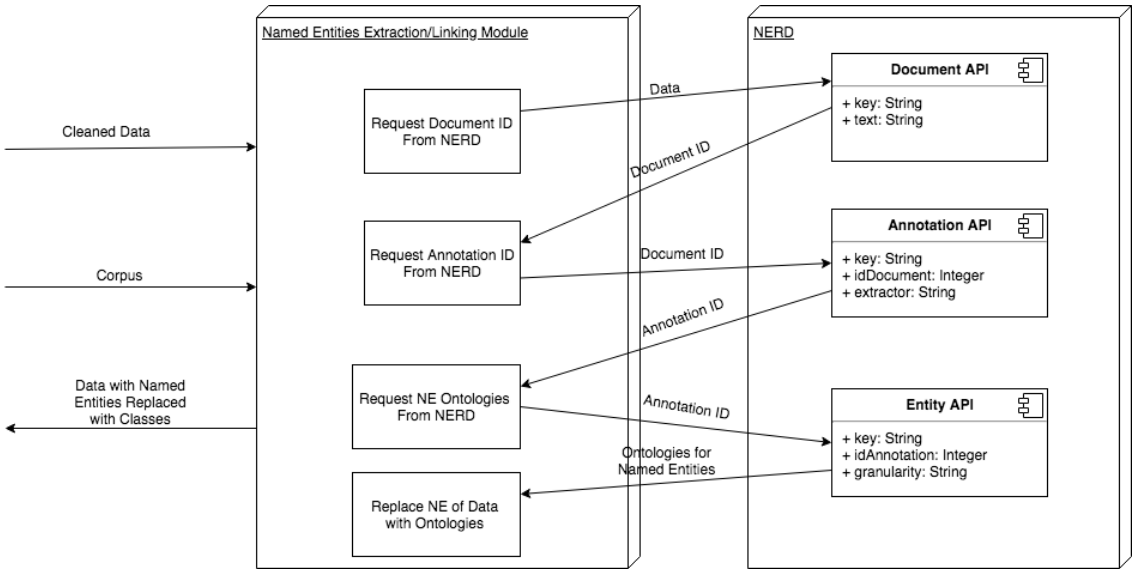


Figure 2: NERD Named Entities Linking

[15] http://nerd.eurecom.fr/api/document
[16] http://nerd.eurecom.fr/api/annotation
[17] http://nerd.eurecom.fr/api/entity

## 5.3 Feature extraction

We represent tweets through term frequency and inverse document frequency (TF-IDF), which have shown a good generalization power, outperforming feature models such as Bag of Words in many NLP tasks. We tokenized tweets in text form to words then created vectors by term frequency for those words.

**Term Frequency/Inverse Document Frequency** [8] is statistical measure which determines how important a particular word is relative to whole corpus. It can also be used to weight information in text mining problems. Its value is directly proportional to the number of times a word appears in corpus. TF-IDF is the dot product of term frequency and inverse document frequency.

However for event detection, time also plays an important role as a feature apart from term frequency, as it is highly possible that event related tweets are clustered together in time. Therefore, in our experiments we used term frequency as well as timestamps of tweets to make feature vectors. Which were then fed to classification algorithms.

Before building features using term frequency, we first cleaned our data using preprocessing steps discussed in section 5.1 then named entities were replaced with classes they belong to as discussed in section 5.2. We created 3 sets for feature vectors:

1. Named entities were replaced by their specific category
2. Named entities were replaced by their general category
3. No named entities were replaced

## 5.4 Tweet Classification

For classification, we used supervised machine learning techniques. Supervised machine learning is a task of developing a model from labeled training data. This training data is used to estimate a mapping function $f$ in such a way that when a new input data $x$ is given to function, it will predict the output variable (class) $y$ for that data input.

$$y = f(x)$$

We will be using some of the common classifiers such as Naive Bayes, Support Vector Machine and Neural Networks.

### 5.4.1 Naive Bayes Classifier

The Naive Bayes classifier [9] is based on the Bayesian theorem [1] with strong independence assumptions between features and is particularly beneficial when the dimensionality of the input is high. It is a highly scalable, fast and simple classifier used for many classification problems. Bayes theorem provides a way of calculation posterior probability from the prior probability of class, the prior probability of predictor and the probability of predictor given class. Naive Bayes classifier assume that the effect of the value of a predictor $x$ on a given class $c$ is independent of the values of other predictors.

$$P(c|x) = \frac{P(x|c) \times P(c)}{P(x)}$$

where $c$ and $x$ are events and $P(x) \neq 0$

### 5.4.2 Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm which is used both for classification and regression problems. However its more commonly used for classification problems where it minimizes the classification error. SVM is based on the idea of finding a hyperplane that best divides a dataset into two classes. In other words, it is a discriminative classifier formally

defined by a separating hyperplane. The classifier builds a model, assigning each data item to one of the categories and then represents the data points in space, mapped in such a way that a clear separation is visible between data points of each categories. Similarly each data point of the test set will be mapped into the same space by predicting their category based on which side of gap they would fall. For linear SVM, a training set of n points will have the form:

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2)...(\vec{x}_n, y_n)$$

where $\vec{x}_1, \vec{x}_2...\vec{x}_n$ are vectors and $y$ is the class to which a particular vector belongs. The hyperplane region can be given by:

$$\vec{w}.\vec{x} - b = 1$$
$$\vec{w}.\vec{x} - b = -1$$

### 5.4.3 Artificial Neural Network

Neural networks are machine learning algorithms based on how neurons in human brain functions: simple computational units are linked together, forming several connected layers to handle a classification task.

Each computational unit is called a neuron (or node). It takes several inputs $x_i$ from the previous layer, each with a specific weight $w_i$. Each neuron has an associated activation function $f$, which is applied to the weighted sum of all inputs, with the addition of a bias $b$ (an additional value). The neuron then transmits the result $y$ to the next layer. A neuron is shown in figure 3.

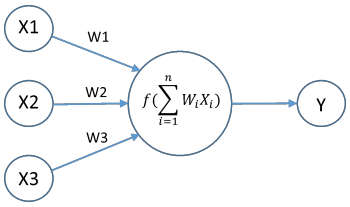


Figure 3: A neuron

A basic neural network architecture would contain an input layer, an output layer, and possibly one or several hidden layers, as shown in figure 4. Each layer can have a different amount of neurons. Neurons within a layer are not linked to each other but neurons are connected between layers.

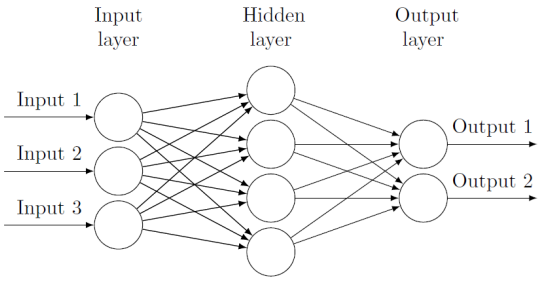


Figure 4: A fully-connected Neural Network with one hidden layer

When an input is given, the network processes it using the weights and functions of its neurons, until an output is obtained. This output can then be compared to an expected output (which represents the truth), and weights are then adjusted using the concept of back-propagation of the error. The general idea is to compute the difference between the expected output and the network's output, which is the error. Then, the weights are modified to reduce this error, and this process is repeated until the error is considered low enough.

### 5.4.4 Random Forest

Random forest [10] is a supervised machine learning algorithm that operates by making large number of decision trees on input data at the time of training. It can be used both for classification and regression problems. Random forest algorithm constructs multiple decision trees and merge them together at the time of output. It is better than decision trees classifier because of the fact that it does not over-fit to training data.

Due to similar input parameters with decision trees, random forest algorithm generate multiple decision trees by splitting a node by the best feature among random subset of features. This results in a better model.

## 5.5 Evaluation

For evaluation of our classifiers, we are using confusion matrix to calculate accuracy, precision, recall and F1 measure for each of the classifier.

### 5.5.1 Confusion Matrix

The confusion matrix (or error matrix) is a way to calculate the performance of a classifier for binary classification tasks. It is a (2x2) square matrix such that for each class the number of true positives, true negatives, false positives and false negatives were calculated as shown in figure 5. Then confusion matrices for each class is combined by taking average of all to create one matrix (also called macro average measure)[4].

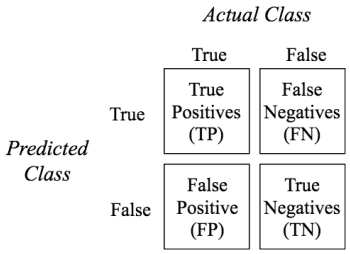


Figure 5: Confusion Matrix

After calculation of confusion matrix, precision measure, recall measure and F1 measure is calculated along with accuracy of classifier.

**Precision** is the ratio of observations predicted as positive to all observations that are positive.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

**Recall (Sensitivity)** is the ratio of observations predicted as positive to all observations that are positive in actual class.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

**F-1 Measure** is the weighted average of Precision and Recall. F-1 score is sensitive to both false positives and false negatives.

$$\text{F-1 measure} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

**Accuracy** is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

$$\text{Accuracy} = \frac{\sum \text{true positives} + \sum \text{true negatives}}{\sum \text{total population}}$$

# 6 Results, Discussion and Conclusion

Table below shows results for precision, recall and F-1 measure for the classifiers we used.

| | NB | | | SVM | | | NN | | | RF | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Approach | Prec | Rec | F1 | Prec | Rec | F1 | Prec | Rec | F1 | Prec | Rec | F1 |
| No entity linking | 89.15 | 42.78 | 57.81 | 89.65 | 69.88 | 78.54 | 12.48 | 25.0 | 16.65 | 91.0 | 71.4 | 80.0 |
| dbp:generic | 89.51 | 48.13 | 62.60 | 90.91 | 69.79 | 78.78 | 65.02 | 63.87 | 64.44 | 90.64 | 77.38 | 83.48 |
| dbp specific | 89.21 | 44.62 | 59.48 | 90.42 | 68.84 | 78.35 | 28.87 | 44.86 | 35.14 | 91.75 | 73.59 | 81.67 |

Table 2: Results Table

As represented by the table 2, we observe that precision, recall and F-1 measures are lowest for the data without linking named entities with their recognized classes. In case of data with named entities replaced, the results for data which have named entities replaced with most generic classes are better than the data having its named entities replaced by most specific classes. Hence we can easily conclude that semantic linking of named entities cast affect on the improvement of results. Below is a visual bar chart representation of results from different classifiers using three types of datasets.

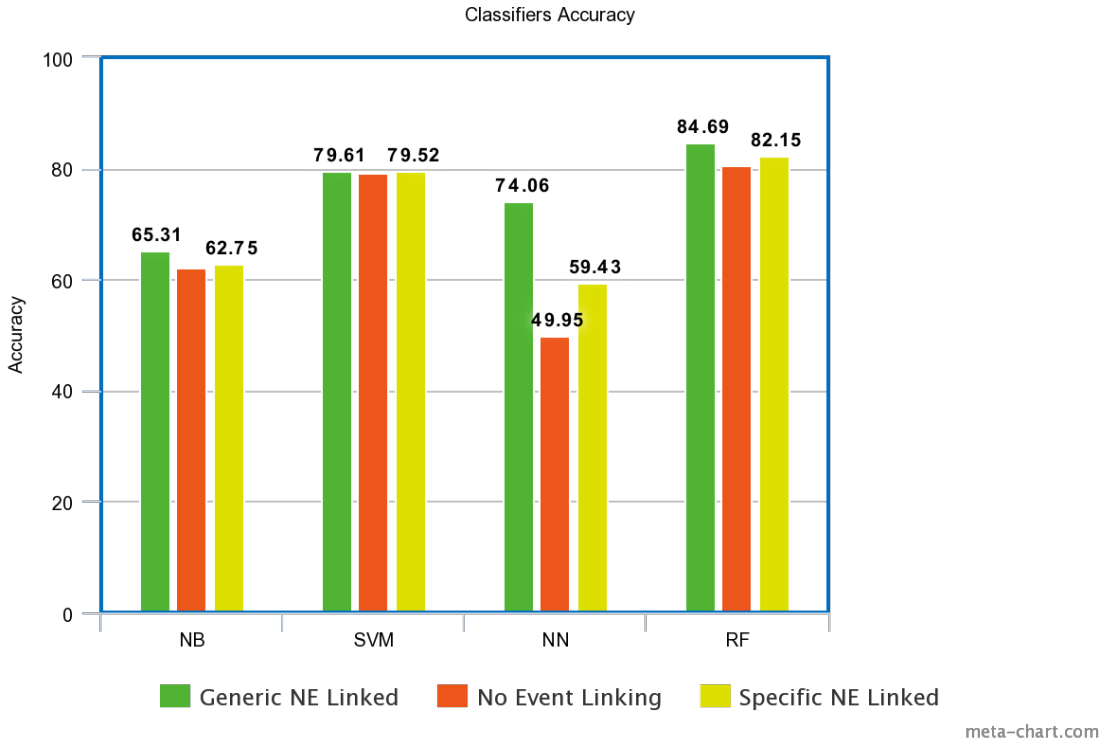


Figure 6: Accuracy Bar Chart

## 7 Future Work

In this project we worked with three different events. In future we would want to include some more types of events in order to get better predictions of classifiers. In the scope of this project, we used four different classifiers, we can later check results of other classifiers e.g. SGD, Logistic Regression, and Decision Tree Classifiers. For named entities replacement, we used NERD for recognizing and linking named entities with their classes, we wanted to use ontologies from YAGO which relatively have more number of defined classes for named entities, but due to limitation on number of requests to YAGO from our account we were not able use its APIs to extract all the named entities. For future, we can use ontologies from YAGO to link named entities in our dataset and check the results for the data with named entities replaced with generic and specific classes retrieved from YAGO.

## References


[1] HS Stern DB Rubin A Gelman, JB Carlin. Bayesian data analysis.

[2] A. Goker R. Skraba S. Papadopoulos D. Corney C. Martin G. Petkos L. M. Aiello. A. Jaimes, I. Kompatsiaris. Sensing trending topics in twitter. ieee transactions on multimedia (pre-print). 2013.

[3] Houben GJ. Tao K. Abel F., Gao Q. Semantic enrichment of twitter posts for user profile construction on the social web. *ESWC 2011*, 2012.

[4] Scholz M. Forman G. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *SIGKDD Explorations*, 2010.

[5] Raphaël Troncy Giuseppe Rizzo. A framework for unifying named entity recognition and disambiguation web extraction tools. *13th Conference of the European Chapter of the Association for computational Linguistics*, 2012.

[6] Sebastian Hellmann Martin Bruemmer Giuseppe Rizzo, Raphaël Troncy. Lifting nlp extraction results to the linked data cloud. *In ( WWW'12) 5th Workshop on Linked Data on the Web (LDOW'12)*, 2012.

[7] Raphaël Troncy Marieke van Erp, Giuseppe Rizzo. Spotting named entities on the intersection of nerd and machine learning. *3rd International Workshop on Making Sense of Microposts*, 2013.

[8] Ullman J.D. Rajaraman A. "data mining" mining of massive datasets. 2011.

[9] Norvig Peter Russell Stuart. Artificial intelligence: A modern approach (2nd ed.).

[10] Ho Tin Kam. Random decision forests. proceedings of the 3rd international conference on document analysis and recognition. 1995.


# Appendices

## A Implementation

Our code can be found in the following repository:
https://bitbucket.org/mzohaibkhan/text-mining