CS 583 - Data Mining and Text Mining

Twitter Sentiment Analysis
Project Report

Project members

Murugappan Nachiappan (656694298)
Gowdhaman Sadhasivam (653746541)

The University of Illinois at Chicago

Spring 2015

Table of Contents

1 Abstract

Evolution of social networks has major impact on human communications where people can express their opinions. Eventually, their opinion can be analysed and result obtained helps to make business decisions and predict future results approximately. In social networks like Twitter, people tweet their views, provide suggestions, opinions about certain topics and chat with other people as well. Tweets about particular topic can be extracted and analyzed using available algorithms, analysed data provides big picture about the topic. In this project we analyze tweets about US presidential election candidates Obama and Romney to predict percentage of candidate supporters.

2 Introduction

2.1 Problem statement

Given collection of tweets about US presidential election candidates Barack Obama and Mitt Romney, we need to classify the tweets such as if a tweet is positive, negative. The input training tweets data have positive, negative, neutral and mixed case with labels 1, -1, 0 and 2 respectively. Mixed case tweets are not considered as per project requirement.

2.2 Approach

Input datasets have noise data and they are removed using pre-processing steps which is described below. In this project two different classification algorithm are used, given below, tweets classification.

- SVM
- Naive Bayes Classifier

Input to the algorithm is MS excel files that consists of tweets sheet of each candidate which has tweets and classes/labels,

- 1: positive class - Tweeter supports candidate
- -1: negative class - Tweeter does not support candidate
- 0: neutral class - Tweets are objective sentences
- 2: mixed class - Tweets has opinion about both candidates or may have both positive or negative opinion about a candidate.

The algorithm uses training sheet to train classifier and classifier uses tweets from test sheets to classify them either positive, negative or neutral. Build classifier is validated by 10-fold cross validation using training data sets.

Programming language used in this project is **Python**. In built Python and **NLTK libraries** are used in pre-processing tweets and building classifier model.

3 Technique

3.1 Data Pre-Processing

Data input to the algorithm is processed to ensure that they are in correct input format and noise data are removed using following steps. The order of pre-processing has impact on input data, following pre-processing order is maintained for both test and training data.

Pre-processing steps:

1. **URL removal** - Removed URL from tweets. URL does not provide enough information and it is not considered as features
2. **Username removal** - User name in tweets starts with @ symbol, word that starts with @ symbols is considered as user name and it is removed. For example, @HiData is a tweeter user name and it is removed from tweets.
3. **Additional white space removed** – Apart from removing noise data in above steps, additional white spaces may be introduced and it is considered as noise. These spaces are removed from tweets.
4. **Extract abbreviations and emoticons** - Tweets have abbreviated words and emoticon symbols. A custom file that consists of most popular tweets and emoticons symbols were used as a lookup table to replace tweet abbreviations and emoticons with meaningful words. For example, symbols like :) , :], :o) are positive smiley, :(, :o(, :'(are negative smiley, :| is neutral. Abbreviations like gr8 and gud are short form of great and good respectively.
5. **Remove Punctuation symbols** - Punctuation symbols that are used to emphasize the tweets are not considered as features and they are removed. For example, 'Awesome!!!' will be corrected as ‘Awesome'.
6. **Remove '#' symbol from #Tags** - Hash tags are prefixed with '#' character which indicates tweets topic. Hash tags are preserved and consider as features, only '#' character is removed from hash tag words. For example, #SupportObama will be SupportObama.
7. **Convert camel-case words** - In tweets and in hash tags camel-case words are used and they are converted to normal words with space separated. For example, SupportObama will be converted as 'Support Obama’.
8. **Convert tweets to lower case** - Capital letter words in tweets were converted to lower case
9. **Remove HTML tags** - Tweets may have HTML anchor and emphasize tags, these tags are removed. For example, tags like <a> , , <e> and </e> are removed from tweets.
10. **Remove symbols** - Any non-letter or symbols that were not removed from tweets during previous steps are removed. For example, smiley images and symbols like, &, * are removed in this step.
11. **Remove Stop words** - Stop words such as a, this, that, there, etc gives no information about tweets. Hence they are removed from tweets. NLTK.CORPUS *stopwords* dictionary is used to check tweets words and removed if they are in stop words dictionary.
12. **English words Extraction** - Words that are not found in English dictionary which are considered as meaningless words and they are removed from tweets. *synsets* library from WORDNET are used to achieve this step. Before extracting meaning full words, words of length less than size two are also meaningless hence they are removed before checking in English dictionary.
13. **Stemming** - Root word of words present in tweets were extracted with help of *stem()* function from *stemming.porter2* library. For example, words like 'voting' and 'voted' have root word 'vote'. These words are replaced with root word 'vote'.

3.2 Implementation

Input data sets are read from excel file and they are shuffled randomly before they are ready for pre-processing. The skewness of data sets are avoid with shuffling that helps to train the classifier with distributed number of samples. Random shuffle will generate different type of deck during each run, seed to random generator is set with constant number. Seed number makes generator to generate same

type of deck during each run. Shuffled data set is sent to pre-processer. The clean data from above processed steps were given as input to algorithm. Both algorithm classifier is tested by 10-fold cross validation before it is used to classify test data sets.

3.2.1 Naive Bayes Classifier

In Naive Bayes Classification algorithm, an additional step is needed to get features set which is extracted from training and test tweets using NLTK *apply_features()* function. Using training feature data, classifier model is developed. Similarly from test data, features set are extracted from tweets and given as input to modelled classifier to classify. Classifier Accuracy, Precision, Recall and F-Score are calculated using NLTK libraries in-built functions.

3.2.2 SVM Classifier

Linear SVM classification algorithm is used as an alternate classifier. Both shuffled training and test data sets are pre-processed using above pre-processing steps. Feature are extracted from processed tweets in vector format using *CountVectorizer()* function of *sklearn* library. In addition to this, TF-IDF value for each feature vectors are calculated and classifier model is modelled with TF-IDF vector of training tweets and its respective sentiment label. Similarly test data tweets are converted into IT-IDF vector and given as input to SVM classifier to classify tweets. Model Accuracy, Precision, Recall and F-score are calculated using NLTK libraries.

4 Evaluation

Output from SVM classification and Naive Bayes classification algorithms are given below.

4.1 SVM Classifier

Linear SVM	OBAMA	ROMNEY
Positive Precision	58.51%	50.65%
Positive Recall	46.61%	42.39%
Positive F-Score	51.89%	46.15%
Negative Precision	69.07%	74.09%
Negative Recall	74.09%	78.38%
Negative F-Score	71.49%	76.17%
Accuracy	58.25%	66.19%

4.2 Naive Bayes Classifier

Naive Bayes	OBAMA	ROMNEY
Positive Precision	61.19%	55.38%
Positive Recall	34.45%	37.89%
Positive F-Score	44.09%	45.00%
Negative Precision	64.64%	72.73%
Negative Recall	82.27%	80.31%
Negative F-Score	72.40%	76.33%
Accuracy	60.10%	63.57%

5 Conclusion

The classification was done using given training and test data sets. It is clearly visible by comparing Liners SVM classification algorithm with Naive Bayes classifier, performance of SVM is better compare than later one. However, high accuracy could not be achieved. As future work, implementing topic model using hash tags, spell checker and auto correcting words in tweets, considering synonyms and antonyms for words in tweets and implementing combination of multiple algorithms using bagging technique can improve the result.