



May 2018

# Twitter Sentiment Analysis

Deep Features team



## Team

Eslam Samy Hosney  
Saad Elsayed Saad  
Mohamed Hassoubah

## Contents

Introduction	2
• Context	2
• Problem	2
The Project	2
1. Data	2
2. Baseline experiments	2
3. After baseline experiments	3
4. Resources	4
5. Features	4
1.5.1 Emoticons	4
1.5.2 Positive and negative words counts	5
1.5.3 Special characters	5
1.5.4 POS sequence.	5
As proposed by Fare Koto, and Mirna Adriana in their paper about POS sequence, the idea is to use POS sequence to represent the tweet then apply sequence matching and get the top 100 repeated sequence, when Len of sequence = 3 accuracy didn't increase but when trying with Len =2 the overall accuracy was increased.	5
1.5.5 POS features	5
6. Pre-processing	5
1.6.1 Emoticons	5
1.6.2 URLs	5
1.6.3 Unicode	6
1.6.4 HTML entities	6
1.6.5 Negation	6
Machine Learning	6
1. KNN	6
2. Naïve Bayes	6
3. Support vector machine	6
4. XGBoost	6
5. Logistic Regression	6
6. Grid search.	6
Task Distribution	7
References	7

## Introduction

- **Context**

This project has been done as a part of our course for the MSc Informatics at Nile University. Supervised by Prof. Dr. Samhaa R. El-Beltagy, We had almost 1 months to fulfill the requirements in order to succeed the module.

- **Problem**

Sentiment analysis, also refers as opinion mining, is a sub machine-learning task where we want to determine which the general sentiment of a given document is. Using machine learning techniques and natural language processing NLP we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral or negative.

In this project, we will try to classify tweets from Twitter into "Positive", "Neutral" or "Negative" sentiment by building a model. Twitter is a website where people can share their feelings quickly by sending tweets limited by 140 characters.

## The Project

1. **Data**

We used annotated data set thousands of tweets were captured in 2013, 2015 and 2016. The dataset originally is a part of kaggle competition SemEval task 4. The tweets are general not related to specific topic. Each tweet was annotated by the sentiment it delivers if it's positive or negative or neutral.

While exploring the data we noticed that the negative samples were almost  $\frac{1}{3}$  the positive samples so we tried to add additional negative tweets but the accuracy plunged.

2. **Baseline experiments**

- 1.2.1 We included the 3 data sets (2013, 2015 and 2016 tweets data sets) in one .csv file. Then we fitted a Countvectorizer to the data. We used 80% of the data as training data to train the logistic regression model and the remaining as validation data. Without any preprocessing done on the data we got accuracy on the validation data = 65.15 %

```
In [49]: tweets_train,tweets_val,train_labels,val_labels = train_test_split(final_train.drop(["Type"],axis=1),final_train['Type'],test_size=0.2)
clfr = LogisticRegression()
#clfr = naive_bayes.MultinomialNB()
#clfr = svm.SVC(kernel='linear')
clfr.fit(tweets_train,train_labels)
predicted = clfr.predict(tweets_val)
acc = metrics.accuracy_score(val_labels,predicted)
print('accuracy = ',acc*100,'%')

accuracy = 65.8234383067 %
```




```
In [50]: print(classification_report(val_labels,predicted))
```





	precision	recall	f1-score	support
negative	0.56	0.40	0.47	289
neutral	0.67	0.76	0.71	932
positive	0.67	0.63	0.65	716
avg / total	0.65	0.66	0.65	1937

It worth mentioning that negative class has the lowest precision and recall and that's because the data is biased toward the other two classes.

- 1.2.2 We removed the stop words then redid the vectorizing and classification we found the accuracy decreased to be 62%.
- 1.2.3 We tried the stemming on each of the tweets, we found that the accuracy decreased to be 52%.
- 1.2.4 We tried multiple classifiers like Naive bayes , SVM , Logistic regression and random forest , but the Logistic regression gave the best results.
- 1.2.5 We tried the TF-IDF vectorizer but it performed less than the Countvectorizer.

### 3. After baseline experiments

Experiment	Impact on accuracy	Conclusion	Discarded
replacing +ve words with the word "positive" replacing -ve words with word "negative"	 by 10%	this will confuse the system specially for sarcastic tweets	TRUE
Removing stop words	 by 2%	Stop words can help in defining the sentiment	TRUE
Replacing emoticons ex. :D, :), :(... with the correspondent vocabularies ex. laughing, happy, sad..	 by 1%	this made it easy for the emotions to be vectorized	FALSE

Removing URLs, usernames (mentions) , numbers and special characters	 by 2%	to avoid adding unnecessary vectors that confuses the classifier	FALSE
Adding the count of +ve words and -ve words for each tweet as 2 vectors	 by 2%	These vectors makes it easier for the classifier to define the sentiment	FALSE
Using only one vector that represents the ratio of the +ve words to the total +ve and -ve words	 by 1%	using a feature for count of the -ve words improves the classifiers	TRUE
Replacing the Countvectorizer with TF-IDF vectorizer	 by 2%	Countvectorizer works better with linear SVM than TF-IDF vectorizer	TRUE

#### 4. Resources

In order to facilitate the pre-processing part of the data, we introduced 4 resources

- Positive-words  
Lexicon that contains words with positive polarity “out of context”
- Negative-words  
Lexicon that contains words with positive polarity “out of context”
- Positive Emoticons  
Lexicon that contains emoticons with positive polarity with their meaning ex, :=), smile
- Negative Emotions  
Lexicon that contains emoticons with negative polarity with their meaning ex, :=(, sad

#### 5. Features

##### 1.5.1 Emoticons

Count both positive and negative emoticons separately and use each as a feature using 2 lexicons which were built by our team.

### 1.5.2 Positive and negative words counts

Using both lexicons provided with the project description, we counted positive and negative words in each tweet

### 1.5.3 Special characters

Count each special character ex '\$percentage&#@' and use as feature.

### 1.5.4 POS sequence.

As proposed by Fare Koto, and Mirna Adriana in their paper about POS sequence, the idea is to use POS sequence to represent the tweet then apply sequence matching and get the top 100 repeated sequence, when Len of sequence = 3 accuracy didn't increase but when trying with Len =2 the overall accuracy was increased.

### 1.5.5 POS features

Using POS tagging we used to count each tag with in the tweet and use it as a feature ex. Counting the number of nouns , counting the number of verbs , counting adverbs and adjectives.

### 1.5.6 Some additional features :

1. Count the number of URLs.
2. Count the number of dots.
3. Count the number of exclamation marks.
4. Count the number of question marks.
5. Count the number of uppercase.
6. Count the number of hashtags.

### 1.5.7 Word embedding

We have tried multiple models ex. Google news and GLOVE. The best results were GLOVE with 200 dimensions. In this feature, we take the average of all words vectors in a tweet

## 6. Pre-processing

### 1.6.1 Emoticons

Using the Emoticons lexicon we have, we replaced each emotion in the tweets to the responding word to keep the same meaning.

Ex: :) -> smile

### 1.6.2 URLs

Using regular expression and matching "http" we were able to remove all the rules.

### 1.6.3 Unicode

During data exploration we found two Unicode characters u2019 and u002c each with replace with the corresponding character in order to keep the meaning and for Emoticons extraction

### 1.6.4 HTML entities

Using beautiful soup library we were able to remove all the HTML words inside each tweet.

### 1.6.5 Negation

The idea was to replace the word after any negation expression (no, not, never,) that lies in the same sentence. To save the meaning of the line using WordNet antonyms function.

Ex. I am not happy -> I am unhappy

However, after testing the overall result seems to decline after applying this preprocessing.

## Machine Learning

### 1. KNN

Worst results among all classifiers.

### 2. Naïve Bayes

Used in baseline testing, fastest algorithm.

### 3. Support vector machine

Produced very good results supported with Kernel type linear and with the data vectored using a countvectorizer.

### 4. XGBoost

Gave the exact same result as logistic regression but too bad performance take more than 10X logistic regression time

### 5. Logistic Regression

Overall considered the best algorithm in terms is performance and accuracy, the one we used for our final code

### 6. Grid search.

Used with all the classifiers mentioned above for hyper parameters tuning.

## Task Distribution

1. Mohammed Mostafa Hassoubah
  - a. Read and summarized the paper “S16-1001” to his mates.
  - b. Generated the best baseline with accuracy 65.15% using Logistic regression.
  - c. Introduced the the (1,2) n-grams as a trial to identify the importance of the word in the context of the tweet.
  - d. Discarded the removal of the stop words and stemming, due to the bad influence on the accuracy.
  - e. Tried adding features ex. counting the number of positive and negative words, use only the ratio of the positive words.
2. Saad El Sayed Saad
  - a. Working on tweet preprocessing and cleaning.
  - b. Handling word tokenization and stemming part with testing these preprocessing steps on dataset and evaluate its performance.
  - c. Feature extraction ex. counting the number of URL, Counting the number of uppercase , Counting the number of exclamation marks , special character and some other features .
  - d. Testing word embedding and implementing its features.
  - e. Handling POS features ex. counting the number of nouns, counting the number of verbs, adverbs and adjectives.
  - f. Implementing Logistic regression classifier and tuning its parameters using Gridsearch.
3. Eslam Samy Hosney
  - a. Run down and summarize “Another Twitter sentiment analysis” by towards data science.
  - b. Run down and summarize “SENTIMENT ANALYSIS OF TWEETS WITH PYTHON, NLTK, WORD2VEC & SCIKIT-LEARN” by Zalbo.
  - c. Introduced and implemented the “Negation correction” function to eliminate the negation words influence on the tweet sentiment.
  - d. Created Emoticons lexicon , separate it into 2 categories positive and negative and used both as different feature.
  - e. Implemented the POS sequence function with variable len to be used as features.
  - f. Implemented Random forest and XGBoost in classifier selection process.

## References

1. <https://www.kaggle.com/valencar/sentiment-classification-of-tweets>
2. [https://www.cse.ust.hk/~rossiter/independent\\_studies\\_projects/twitter\\_emotion\\_analysis/twitter\\_emotion\\_analysis.pdf](https://www.cse.ust.hk/~rossiter/independent_studies_projects/twitter_emotion_analysis/twitter_emotion_analysis.pdf)
3. <https://github.com/anthonyray/sentiment-analysis>
4. <https://zablo.net/blog/post/twitter-sentiment-analysis-python-scikit-word2vec-nltk-xgboost>
5. <https://towardsdatascience.com/another-twitter-sentiment-analysis-bb5b01ebad90>