

# Twitter Sentiment Analysis using Distant Supervision



## Machine Learning Summer Training 2018

NLP Track

---

Ahmed Walid Kamal

Salma Ahmed Awad

Yassmin Abdel Nasser Barakat

3rd September, 2018

## Problem Definition

Twitter is a popular microblogging service where users create status messages called “tweets”. These tweets sometimes express opinions about different topics. We introduce an approach for automatically classifying Twitter messages to either positive or negative for English and Arabic tweets.

## Approach

Our approach is to use different machine learning classifiers and feature extractors as well as Artificial Neural Networks (ANN). The machine learning classifiers are Logistic Regression, Naive Bayes,, Multinomial NB, Ridge Classifier, Passive-Aggressive Classifier and Support Vector Machines (SVM). The Artificial Neural Network is used along with Tfidf vectorizer The feature extractors are unigrams, bigrams and trigrams. We built a framework that treats classifiers and feature extractors as two distinct components.

## Dataset Description and Preprocessing

### I. Description

The dataset for training, We chose “Sentiment140”, which originated from Stanford University. More info on the dataset can be found from the [link](#). The dataset can be downloaded from the [link](#).

Dataset has 1.6 million entries, with no null entries, even though the dataset description mentioned neutral class, the training set has no neutral class.

50% of the data is with negative label, and another 50% with positive label.

By looking at the description of the dataset from the link, the information on each field can be found.

- 0 — the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
- 1 — the id of the tweet (2087)
- 2 — the date of the tweet (Sat May 16 23:58:44 UTC 2009)
- 3 — the query (lyx). If there is no query, then this value is NO\_QUERY.
- 4 — the user that tweeted (robotickilldozr)
- 5 — the text of the tweet (Lyx is cool)


## II. Preprocessing

The training data is post-processed with the following filters:

1. Emoticons listed in the below table are stripped off. This is important for training purposes. If the emoticons are not stripped off, then some classifiers tend to put a large amount of weight on the emoticons, which hurts accuracy.

Emoticons mapped to :)	Emoticons mapped to :(
:)	:(
:-)	:-( :(
: )	: (
:D	
=)	

2. Any tweet containing both positive and negative emoticons are removed. This may happen if a tweet contains two subjects. Here is an example: Target orientation :( But it is my birthday today :). These tweets are removed because we do not want positive features marked as part of a negative tweet, or negative features marked as part of a positive tweet.
3. Retweets are removed. This usually happens if a user likes another user's tweet. Retweets are commonly abbreviated with "RT." For example, consider the following tweet: Awesome! RT @rupertgrintnet Harry Potter Marks Place in Film History <http://bit.ly/Eusxi> :).
4. Tweets with ":P" are removed. As the Twitter API has an issue in which tweets with ":P" are returned for the query ":(". These tweets are removed because ":P" usually does not imply a negative sentiment.
5. Repeated tweets are removed. Similar to retweets, duplicates are removed to avoid putting extra weight on any particular tweet
6. Converting HTML encoding to text
7. Replacing any url with class URL
8. Replacing any @username with class USERNAME

- 
9. Striping repeated chars. For example “Huuuuugry !” becomes “Huungry !”
  10. Replacing #hashtag with hashtag
  11. Removing Numbers

## Methods

### I. Model Building

Before we can train any model, we first consider how to split the data. Here we chose to split the data into three chunks: train, development, test.

- Train set: The sample of data used for learning
- Development set (Hold-out cross validation set): The sample of data used to tune the parameters of a classifier, and provide an unbiased evaluation of a model.
- Test set: The sample of data used only to assess the performance of a final model.

The split ratio was 98% of data as the training set, and 1% for the validation set, and the final 1% for the test set as follows :

- ***Train set has total 1565232 entries with 49.98% negative, 50.02% positive***
- ***Validation set has total 15972 entries with 50.83% negative, 49.17% positive***
- ***Test set has total 15972 entries with 50.08% negative, 49.92% positive***

## II. Feature extraction

We used two feature extraction methods **count vectorizer** and **TFIDF vectorizer**, using different “n-grams” (unigrams, bigrams and trigrams) with and without English stop words in dataset.

The model we chose is logistic regression as it is one of linear models, so it is computationally scalable to big data.

We tried different number of features to make a decision on whether to remove stop words or not, then we get the optimal accuracy at 100k features.

We tested bigrams and trigrams only with English stop words in dataset, as we get higher accuracy using unigrams with stop words.

**Table 1: count vectorizer accuracy**

n-gram	With English stop words	Without English stop words
unigrams	80.49	78.16
bigrams	82.19	—
trigrams	82.26	—

**Table 2:TFIDF vectorizer accuracy**

n-gram	With English stop words	Without English stop words
unigrams	80.66	78.55
bigrams	82.55	—
trigrams	82.73	—

### III. Models

- Classifiers

we used these basic models and compare their validation result using TFIDF vectorizer with trigrams , then we build voting classifier with top five models.

1. Logistic Regression
2. Naive Bayes
3. Multinomial NB
4. Ridge Classifier
5. Passive-Aggressive Classifier
6. Support Vector Machines (SVM)

- Artificial Neural Network

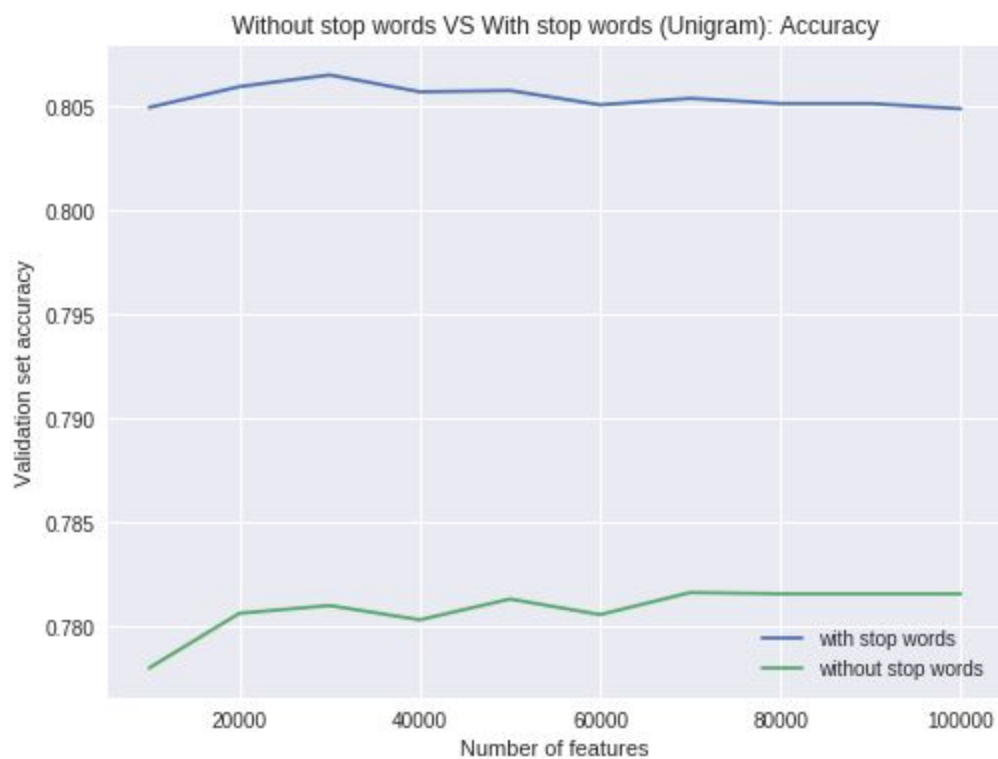
The structure of NN model has **100,000** nodes in the input layer, then **64** nodes in a hidden layer with Relu activation function applied, then finally one output layer with sigmoid activation function applied.

There are different types of optimizing techniques for neural networks, and different loss function you can define with the model. In our model we used ADAM optimizing, and binary cross entropy loss.

We trained our model using 20% drop out of hidden layer with shuffling data for each epoch.

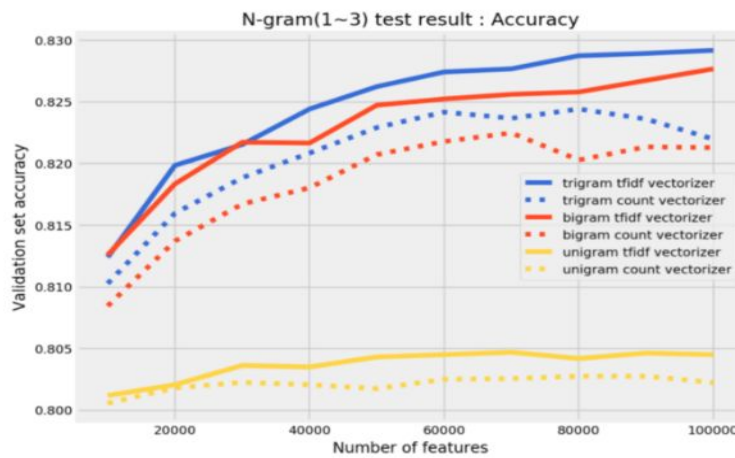
## Results

- **In case of Unigrams** , the evaluation result shows that removing stop words did not improve the model performance, but keeping the stop words yielded better performance.

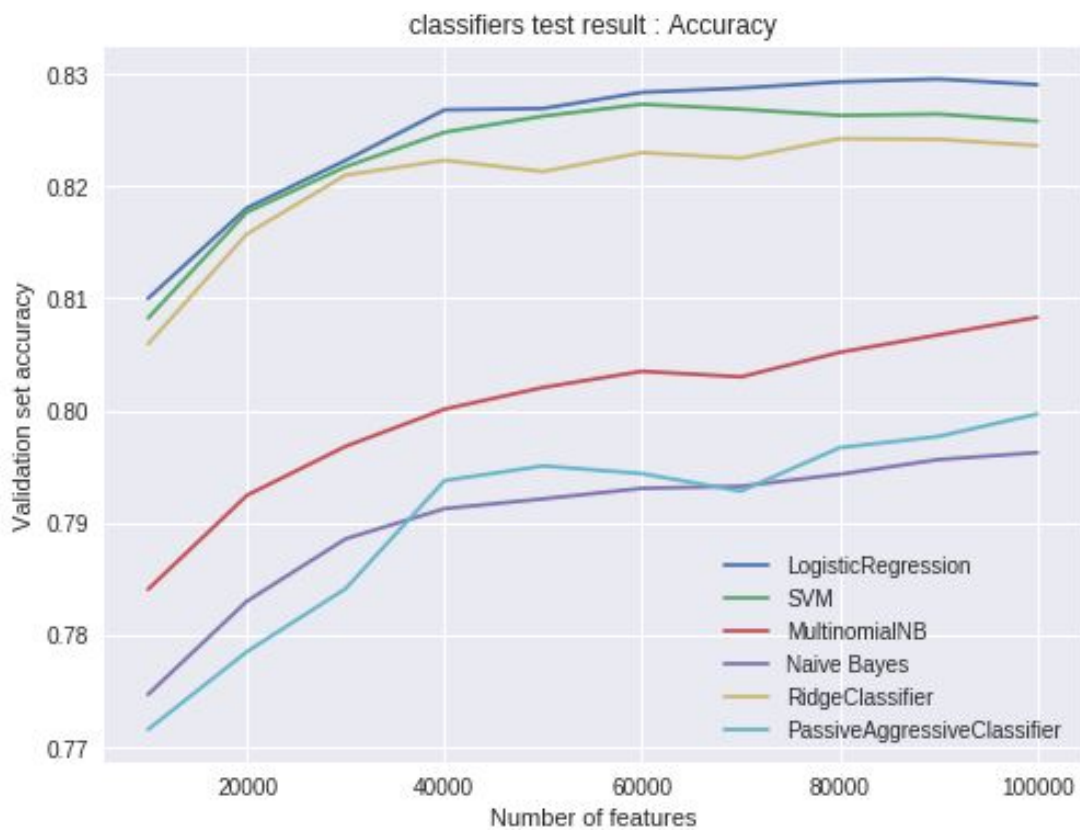





- comparison between count and tfidf vectorizer



- **Logistic Regression** is the best Performing classifier as shown below



- 
- The best result we can get with logistic regression was by using TFIDF vectorizer of 100,000 features including up to trigram .Validation accuracy is **82.73%**, while train set accuracy is **84.18%**
  - Training tfidf vector with a Neural Network of 100k features, using 20% dropout gave validation accuracy of **82.58%** and train set accuracy of **84.26%**

## Conclusion

- using emoticons as noisy labels for training data is an effective way to perform distant supervised learning.
- Logistic regression achieve high accuracy for classifying sentiment.
- Neural Network failed to outperform logistic regression in terms of validation.This might be due to the high dimensionality and sparse characteristics of the textual data.

## Arabic Tweets Sentiment Analysis

### I. Dataset Description and Preprocessing

For the arabic dataset, it was collected using a scraper called “Tweet Scraper” which can be found here [Tweet Scraper](#). The scraper was used to collect nearly 800k of positive tweets and 800k of negative tweets using emoticons as a means to label the tweets.

After collecting the dataset, nearly the same preprocessing steps were applied to the arabic tweets as most of the “cleaning up” needed was to clean properties that are common to all tweets(mentions, hashtags, etc..)

### II. Methods

#### A. Model Building

The data was split the same way as the English dataset was split (98% for the training set, 1% for the development set and 1% for the test set):

- Train set has total 1639816 entries with 48.05% negative, 51.95% positive
- Validation set has total 16733 entries with 48.50% negative, 51.50% positive
- Test set has total 16733 entries with 48.05% negative, 51.95% positive

#### B. Feature Extraction

The same feature extractors were experimented on the Arabic dataset (Count Vectorizer and TFIDF with Logistic Regression as the determining classifier) and the results were as follows:

- Count Vectorizer’s best accuracy was produced using Unigrams with the elimination of the top **30** most frequent words in the dataset (custom stop words), with accuracy = **78.86%**

- TFIDF's best accuracy was produced using Unigrams with the elimination of custom stop words too, with accuracy = **78.03%**

*Notice that the English dataset had better results with the TFIDF feature extractor and when using trigrams while the Arabic dataset had better results with Count Vectorization using Unigrams.*

## C. Models

### 1. Classifiers

Using the Logistic Regression classifier, the best validation accuracy scored was **78.7%** which is lower than that of the English dataset.

### 2. Artificial Neural Networks

Using the same architecture of the ANN used for the English dataset, it produced much better results on the Arabic dataset, scoring nearly **87.6%** validation accuracy which is significantly higher than any accuracy reached in the English dataset

## Challenges

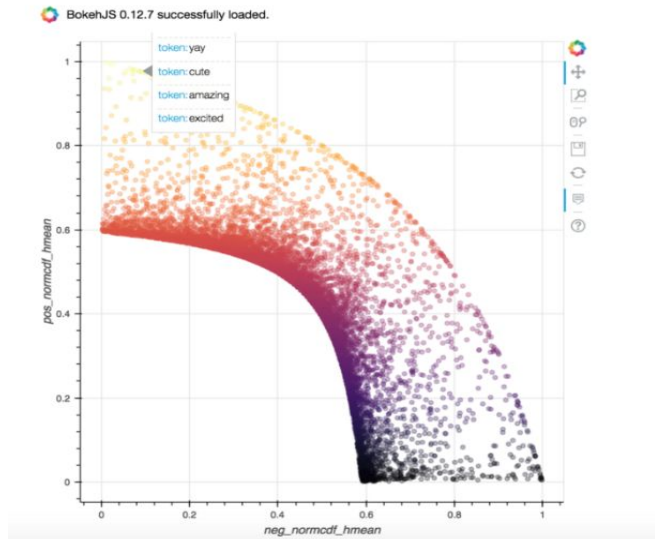
- Arabic Dataset Collection using a twitter scraper where collecting 1.6 million tweets took around 2 days where the scraper had to be restarted several times.
- Encoding problems concerning the files of the dataset before and after preprocessing
- Most of the Visualization and Dataset Analysis Libraries do not support Arabic Language

## Practical tips and tweaks

- **Google Colaboratory Notebook** with GPU as run time was used to accelerate the running of different classifiers and the Neural Network.
- Data was imported into **pandas frameworks** in order to make data processing and analysis easier
- **Scikit learn** libraries was used when dealing with different classifiers.
- **TextBlob** is a Python library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction and sentiment analysis
- **Word clouds** are graphical representations of word frequency that give greater prominence to words that appear more frequently in a source text. The larger the word in the visual the more common the word was in the document(s). This type of visualization can assist evaluators with exploratory textual analysis by identifying words that frequently appear in a text.



- **Bokeh** is an interactive visualization library for Python that enables beautiful and meaningful visual presentation of data in modern web browsers. It provides an insight of data analysis.



- **Keras** Library which is built on top of Tensorflow was used for efficiently modeling the Neural Network.
- A scrapper was used to collect the arabic dataset of tweets which can be found [here](#). TweetScraper can get tweets from [Twitter Search](#).

It is built on [Scrapy](#) without using [Twitter's APIs](#). The crawled data is not as *clean* as the one obtained by the APIs, but the benefits are you can get rid of the API's rate limits and restrictions. Ideally, you can get all the data from Twitter Search.

