# Twitter Sentiment Analysis

Tijana Ilievska
Politecnico Di Torino
s301381
s301381@studenti.polito.it

Sevde Bekdik
Politecnico Di Torino
s298317
s298317@studenti.polito.it

***Abstract*—In this article, we propose a method to predict the sentiment of the tweets with a given textual data on Twitter. The proposed approach consists of a machine learning pipelines involving the use of Linear Regression, Linear Support Vector Classifier and Naive Bayes along with using Term Frequency-Inverse Document Frequency (TF-IDF). F1 score and accuracy are used in evaluating the performance of the classifier.**

## I. Problem overview

Nowadays, through the online communities, large volumes of sentiment rich data is generated in different formats. Sentiment analysis or also known as 'opinion mining' is the process of extracting subjective information which can help in determining the polarity of the data. This project aims to build a classification model to predict the sentiment of tweets on Twitter either positive or negative. The data set is split into two portions:

1) "Development set", collection of 224597 tweets in tabular format and characterized by;
   - *ids*: a numerical identifier of the tweet;
   - *date*: the publication date;
   - *flag*: the query used to collect the tweet;
   - *user*: the username of the original poster;
   - *text*: the text of the tweet;
   - *sentiment*: the sentiment of the tweet which is positive if equals 1 and negative if equals to 0.
2) "Evaluation set", containing 74872 tweets without the *sentiment* feature.

We have already labeled data and our goal is to classify the sentiment of the tweets represented in the *'text'* column of the Evaluation set, by using the proposed models. The models are trained and validated on the Development set.

First of all, we imported the data set and libraries required and did some preliminary exploration of the development data in order to identify possible problems. As we can see from the Fig. 1, the data set is balanced and no **duplicate** or **missing** values were found in it. Next, we performed data preprocessing in order to clean and convert textual data to numeric data, and after that, we split our data into a training set which is used for training the model, and a testing set in order to evaluate the model. Once data is divided, we used three different algorithms to train our model; Linear Regression, SVM, and Naive Bayes. Lastly, after training and testing the model, we used f1-score and accuracy in order to evaluate the performance.

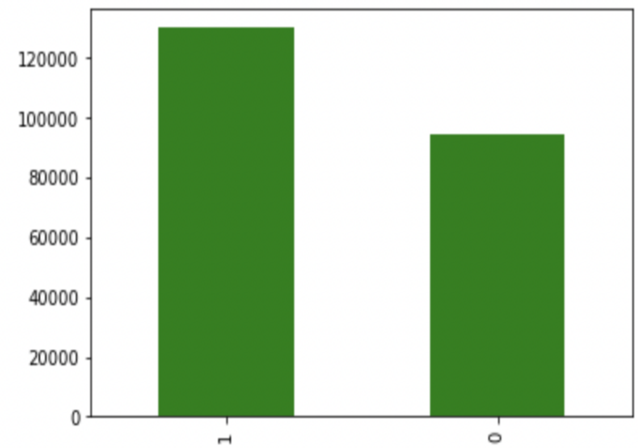


Fig. 1. Distribution of the target variable in the Development Set

## II. Proposed approach

### A. Preprocessing

The first step of our work is concatenating both the development and evaluation set in order to be able to preprocess both sets at once. In addition, we drop the "user", "flag", "date" and "id" columns since we believe that they do not have any effect on what sentiments users are expressing in a certain tweet. Our outcome is based solely on the "text" column, thus we are only using these two columns in the further processing. The raw text can contain a lot of things which are not important for our analysis, such as punctuation, numbers, stopwords etc. Also, one word can have different forms due to grammatical reasons and it is important to normalize the word to its root form. To tackle these issues, we applied the following steps to clean the data:

- Converting all the letters to lower
- Removing all URLs
- Replacing emoticons with their sentiments
- Removing usernames
- Removing any extra empty spaces
- Removing any numbers
- Replacing any three or more consecutive letters with two
- Lemmatization (converting words to their root form)

We also explored additional preprocessing techniques, such as removing stopwords and stemming, but in our case this particular set of preprocessing steps has shown to be the most successful.


Fig. 2. Word Cloud with Negative Sentiment

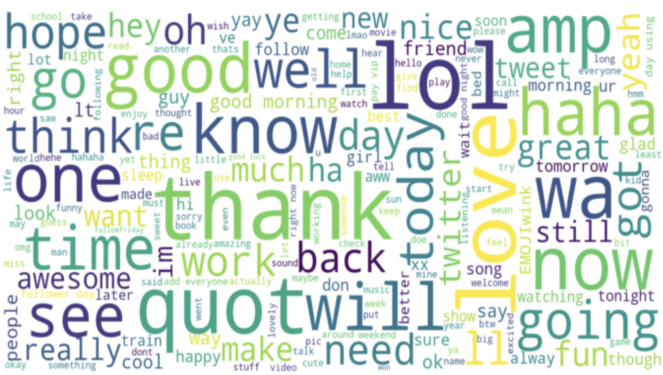


Fig. 3. Word Cloud with Positive Sentiment

After preprocessing, we plotted WordClouds for Positive (1) and Negative (0) tweets from the data set as can be seen in Fig. 2. and Fig.3. We wanted to see and represent which words define the sentiment mostly.

As a next step, and before we split the dataset into test and train, we separated the development and evaluation sets again. This was done by separating the instances that have the *"sentiment'* column and the ones that do not have it. Now that we have the develpment set again, we split it into test and validation sets, with a 25% of the set reserved for validation. From this part on we use the evaluation set as a test set only.

After the cleaning of the tweets and splitting the datasets, we are left with a data with many different aspects that need to be extracted. We used tf-idf (term frequency-inverse document frequency) to put statistical weight on each word and see how important it is in determining the polarity of the data. Tf-idf represents the number of times a given word appears in a reviews relative to the number of reviews in the corpus that the word appears in.

### B. Model selection

This section gives an overview of the classification methods that are used in this paper. After researching about what are the most commonly used algorithms for this kinds of tasks

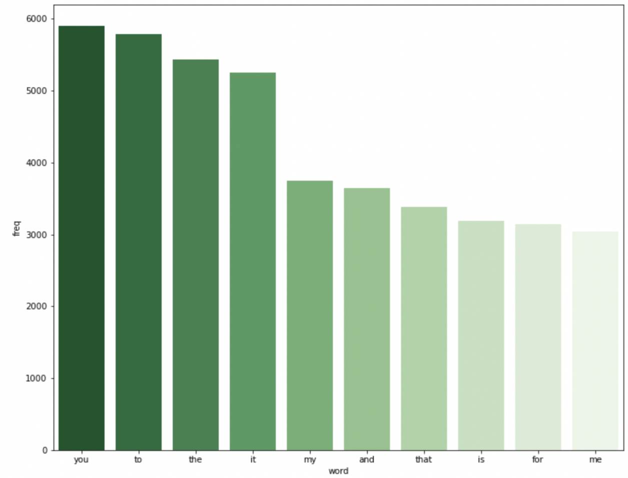


Fig. 4. Frequency of the most common words

and since linear classifiers are more fit to work on this type of sparse datasets,we decided to use the following three and figure out which is the best performing among them:

- Logistic Regression: in our research, we have found Logistic Regression to be very effective on text data and the algorithm behind is fairly easy to understand. It uses a logistic function to model a binary dependent variable.
- SVM: this supervised learning algorithm is based on vector theory and the idea is finding a hyper plane that best divides the data set into two classes. The goal is to generate a hyper plane that creates the maximum margin between the points of the different data sets. SVM is commonly used for text classification since it has shown to perform well.
- Naive Bayes: it is an algorithm based on Bayes' Theorem with the assumption that no pair of features are dependent. The model is simple, easy to build and is considered useful for large data sets.

### C. Hyperparameters tuning

We tried to perform GridSearch to tune the hyperparameters for the models but it in our case it took too long to execute so we decided to use Randomized Search. The development dataset had already been split into train and validation sets previously and we used the validation set to run Randomized Search on it. We kept the default 5-fold cross validation in the search to make sure no overfitting is happening. For each model we considered the following hyperparameters sets:

| Model | Parameter | Parameter values |
|---|---|---|
| **Logistic Regression** | C<br>penalty | { 0.01, 0.1, 1, 2, 5, 10, 100, 1000}<br>{'l1', 'l2'} |
| **LinearSVC** | C | {0.01, 0.1, 1, 10} |
| **Naive Bayes** | alpha | {0.01, 0.1, 0.5, 1.0, 10.0} |

TABLE I
HYPERPARAMETERS CONSIDERED

The results of the search for each model are shown bellow:

| Model | Parameter | Calculated value |
|---|---|---|
| **Logistic Regression** | C<br>penalty | 2<br>l2 |
| **LinearSVC** | C | 0.1 |
| **Naive Bayes** | alpha | 0.5 |

TABLE II
HYPERPARAMTERS TUNING WITH RANDOMIZED SEARCH

## III. RESULTS

By analyzing and comparing all the results after evaluating the model, we are now able to determine the best algorithm in order to make a prediction.

Considering the accuracy of the model, it is observed that the best performing algorithm is Logistic Regression, while SVM follows it and outperforms Naive Bayes as can be seen in Fig. 5.

The f1-score for both positive and negative sentiments:

- For 0: Logistic Regression (f1-score = 0.74) > SVM (f1-score = 0.73) > Naive Bayes f1-score = 0.70)
- For 1: Logistic Regression (f1-score = 0.82) > SVM (f1-score = 0.81) > Naive Bayes (f1-score = 0.80)

Therefore, Logistic Regression is concluded as the best algorithm for the prediction.

| Model | f1-score |
|---|---|
| Logistic Regression | 0.79 |
| SVM | 0.78 |
| Naive Bayes | 0.76 |

Fig. 5. Accuracy of Model Selection

The confusion matrices of the algorithms with the ratios between predicted and actual values, for both outcomes, are given in Fig. 6, Fig. 7 and Fig. 8.

After fitting the models with the tuned parameters, we had a slight improvement in the performance of all the algorithms but Logistic Regression was still outperforiming the other two.

| Model | Before tuning | After tuning |
|---|---|---|
| **Logistic Regression** | 0.7871 | 0.7873 |
| **LinearSVC** | 0.7790 | 0.7860 |
| **Naive Bayes** | 0.7642 | 0.7646 |

TABLE III
ACCURACY IMPROVEMENT WITH HYPERPARAMETERS TUNING

## IV. DISCUSSION

During our work on this project, we tried different types of preprocessing and what came as a surprise to us was that removing the stop words resulted with lower performance. We attempted this by using both pre-defined list from the nltk

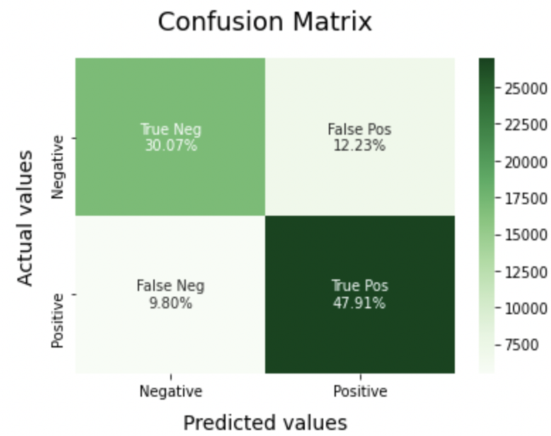


Fig. 6. Confusion Matrix for Linear SVC

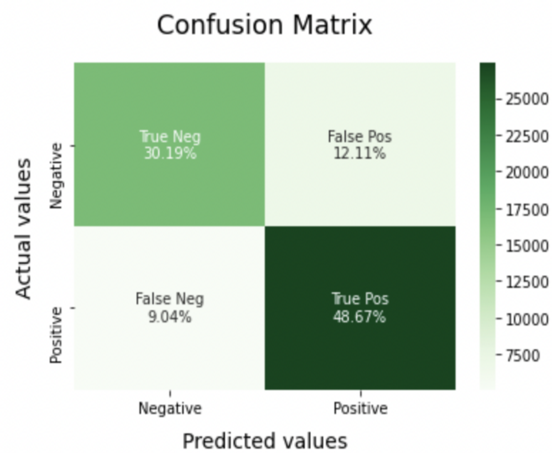


Fig. 7. Confusion Matrix for Logistic Regression

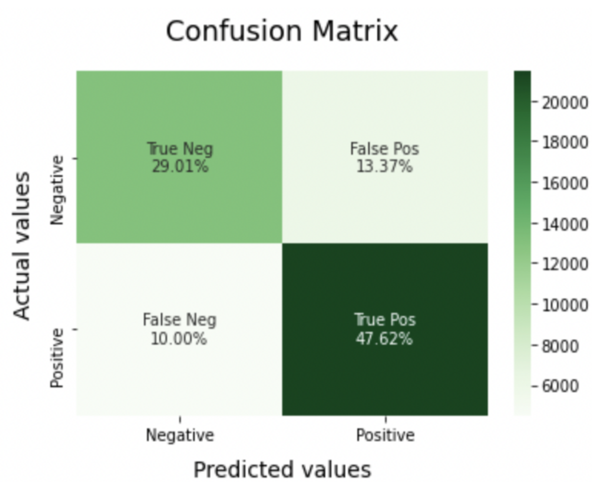


Fig. 8. Confusion Matrix for Naive Bayes

library and a custom list of stopwords (without negation words which are actually present in the nltk library). For the sake of the performance, we decided to keep all the words in the final model. Also, we observed that considering the sentiment of the emoticons in our analysis has shown to considerably improve the overall sentiment score on both positive and negative opinions. Due to large size of the dataset and our limited resources we were unable to perform GridSearch and only used a small subset of random parameters for tuning. We believe that there are better arrangements of parameters that could result in improved performance. After all, we can see that all three models performed with very small differences in the accuracy. Based on this we can conclude that preprocessing is a key factor for effective sentiment analysis.

## REFERENCES


[1] Sampoornam, K. P., Hemalatha B., Kaaviyasri, M., Jaya Priyanka, K, "Sentiment analysis using Machine learning techniques on python", *Conf. Series: Materials Science and Engineering*, doi:10.1088/1757-899X/1084/1/012028 .

[2] Agarwal, A., Xie, B., Vovsha, I.,Rambow, O., Passonneau, R., "Sentiment analysis of Twitter data", *2003 IEEE MTT-S Int. Microwave Symp. Dig.*, vol. 3, pp. 32-33, 2020.

[3] Saif, H., Fernandez, M., He, Y., Alani, H., O., '"On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter"', *Knowledge Media Institute, The Open University, UK*

[4] Scikit-learn: "Machine Learning in Python", *Pedregosa et al., JMLR 12* pp. 2825-2830, 2011

[5] B. Liu, "Sentiment analysis and subjectivity". *Handbook of natural language processing*, 2010.