Twitter Sentiment Analysis

Maria Rosa Scoleri
Politecnico di Torino
Student id: s301841
s301841@studenti.polito.it

***Abstract*—In this report we propose an approach to the sentiment analysis problem on micro-blogging and specifically on tweets. Our methodology collects various preprocessing techniques and applies a TF-IDF-based textual analysis. Information about users and dates are partially maintained and the vectorized text is then used as input for two classification models. This approach outperforms the baseline defined for the problem and returns satisfactory results.**

I. Problem overview

The proposed competition is a classification problem on a dataset containing a collection of tweets. The goal of this project is to perform sentiment analysis and assign to each row of the dataset the correct label ”zero” (for negative tweets) or ”one” (for positive tweets). The dataset is divided into two parts:

- a *development* set containing 224597 rows
- an *evaluation* set containing 74872 rows

We will use the development set to build a classification model to correctly label tweets in the evaluation set. We can make some observations based on the development set starting from the description of the columns present in the dataset:

- *sentiment*: feature that takes either the value 0 for the Negative class or 1 for the Positive class;
- *ids*: numerical unique identifier of the tweet;
- *date*: publication date containing day of the week, month, day of the month, PDT time and year
- *flag*: query used to collect the tweet (is equal to the string ’NO QUERY’ for every row in the dataset)
- *user*: username of the original poster
- *text*: text of the tweet.

Firstly, as shown in Fig. 1, we can see that the development set is not perfectly balanced: positive tweets compose about 58% of the dataset.

Secondly, there are about ten thousand unique users in the dataset, which means that some users wrote more than one tweet. This brings the observation that some users may tend to write either more positive or more negative tweets. The conclusion is supported by the plot shown in Fig.2: we see that a large number of users is associated with a mean sentiment different from 0.5. The figure also highlights that almost 800 users have a mean sentiment equal to 1 which means the only wrote positive tweets.

Going further with the problem overview we can consider dates and, specifically, days of the week. Fig. 3 represents the number of positive and negative tweets per day of the week:

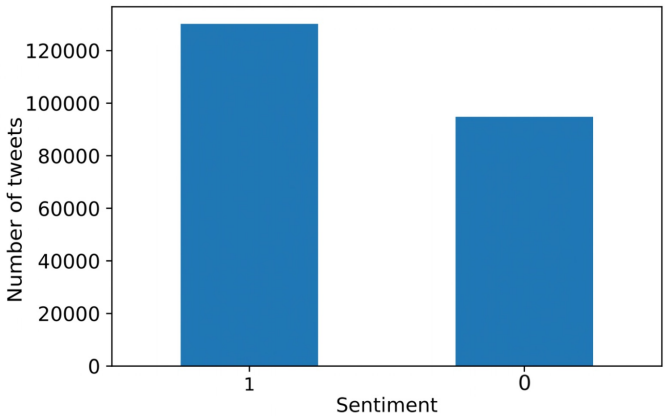


Fig. 1: Number of positive and negative tweets in the *development* set

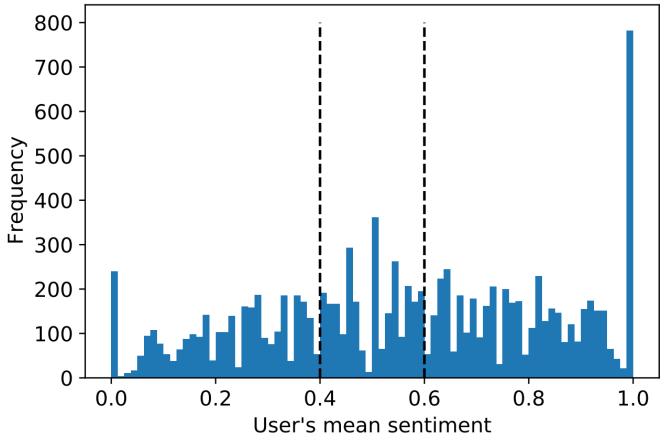


Fig. 2: Frequency of users’ mean sentiment. The bins included between dashed lines represent neutral users: these users write a comparable number of positive and negative tweets.

some days show a higher number of tweets with the label ”zero” and others with the label ”one”.

It is also relevant to know that the dataset does not contain any missing values.

Finally, we can separate positive and negative tweets and visualize words that are more common in one class or the other. Figures 4a and 4b show the word clouds made with TF-IDF for tweets with positive and negative sentiment. The figures allow us to see that several words are common in both classes while others are more characteristic of one class.

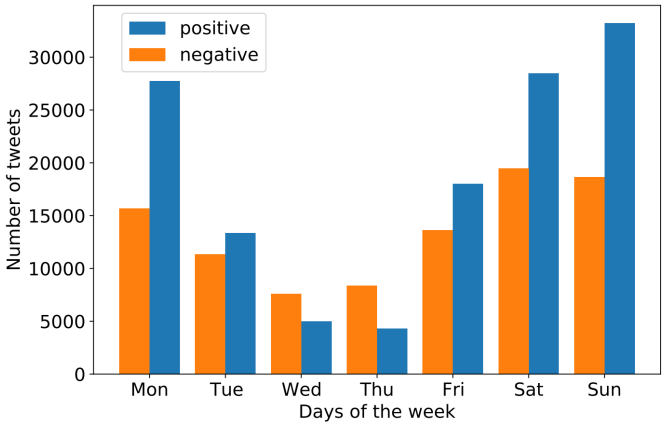


Fig. 3: Number of positive and negative tweets for each day of the week

II. Proposed approach

A. Preprocessing

We have seen that our dataset is not perfectly balanced so we can perform either under-sampling or oversampling. Since the dataset contains a copious amount of rows we can under-sample without experiencing a significant loss of information. Therefore, we randomly select a subset of samples from the class with more instances (label 1) to match the number of rows associated with the label 0.

In the previous section, we showed that several users have a tendency to write either positive or negative tweets. This indicates that usernames could play a significant role in identifying which tweet belongs to each class. We then copy the usernames in the tweets' text so that they will appear as words in our further analysis.

The same reasoning can be applied to days of the week: we noticed that the distribution of positive and negative tweets varies within the week. This drives us to add days names to the text column as we did with usernames.

Columns called *ids,date,flag,user* can now be removed from the dataset since they do not contribute to a better understanding of the sentiment.

Words contained in the 'text' column of the dataset are subject to further preprocessing techniques. Using the Regular Expression library [1] we remove links, mentions, punctuation and numbers since they do not deliver any relevant information for the sentiment analysis. We also remove repeating characters from the words.

The final step of the preprocessing is text vectorization. Specifically, we apply the TF-IDF vectorizer included in scikit-learn [2] with a custom-built tokenizer. The tokenizer removes the vectorization of words shorter than two characters and longer than sixteen. After, it performs the stemming and lemmatization of the text using functions provided in the Natural Language Toolkit library [3]. The TF-IDF vectorizer allows us to pass as a parameter a list of stopwords to be removed from the text. This list needs to be normalized with the same operations already described for the text.

Model	Parameter	Values
Preprocessing	Number of features	500→32000, doubling at every step
Linear SVC	penalty	{'l1', 'l2'}
	dual	{True, False}
	max_iter	{1000, 2000}
	C	{0.1, 1, 5, 10, 100}
Logistic Regression	penalty	{'l1', 'l2'}
	dual	{True, False}
	max_iter	{100, 150, 200}
	C	{0.1, 1, 5, 10, 100}

TABLE I: Preprocessing and model's hyperparameters considered

B. Model selection

Among the models the have been proved to perform the best with sentiment analysis, we decided to test the following:

- *Linear SVC*: implementation of Support Vector Machine classifier using libsvm. It is similar to SVC with linear kernel but we chose to use Linear SVC because of the fastest convergence.
- *Logistic Regression*: classification algorithm that uses the logistic function to compute the probability of an events' occurrence. The logistic function is applied to the output of a linear regression model so that the result becomes binary.

These classifiers were preferred over the alternatives because they proved to be scalable to large datasets, do not require tremendous amounts of memory and have been shown to be effective with sentiment analysis [4].

C. Hyperparameters tuning

There are two main sets of hyperparameters to be tuned:

- *max_features*: maximum number of words that the TF-IDF vectorizer will use as features;
- Linear SVC and Logistic Regression hyperparameters.

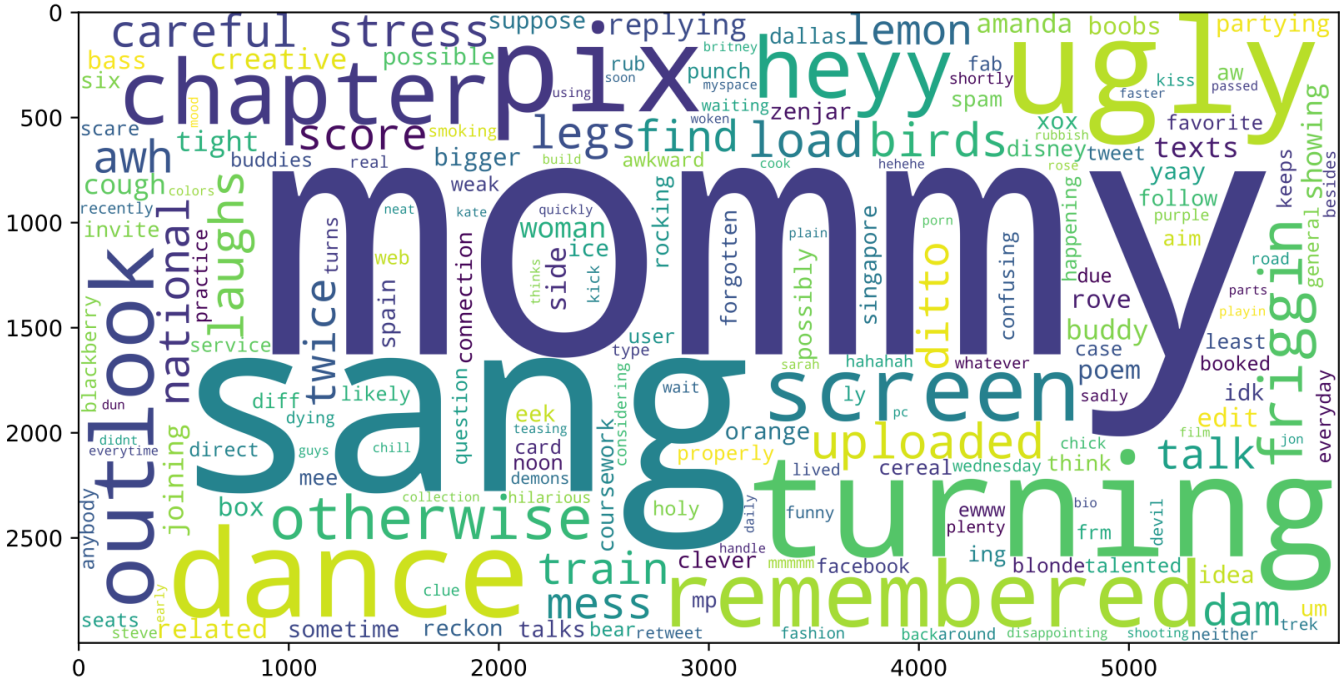
We perform a 70/30 train/test split on the development set and run a grid search on *max_features* with the range of values shown in Tab. 1. To perform this task we use Linear SVC and Logistic Regression with their standard configurations. The resulting *f1* score allows us to assess the performance of both models and to consequently choose a proper value of features to be considered[1]. After finding the best value for *max_features*, we run a grid search on the selected models, based on the hyperparameters shown in Tab. 1.

One more parameter passed as input to the TF-IDF is the list of stopwords. It has been debated whether the removal of stopwords is effective in sentiment analysis [5] so we decided to try both approaches. We train our models with standard configuration and assess their performance both with and without the list as a parameter.

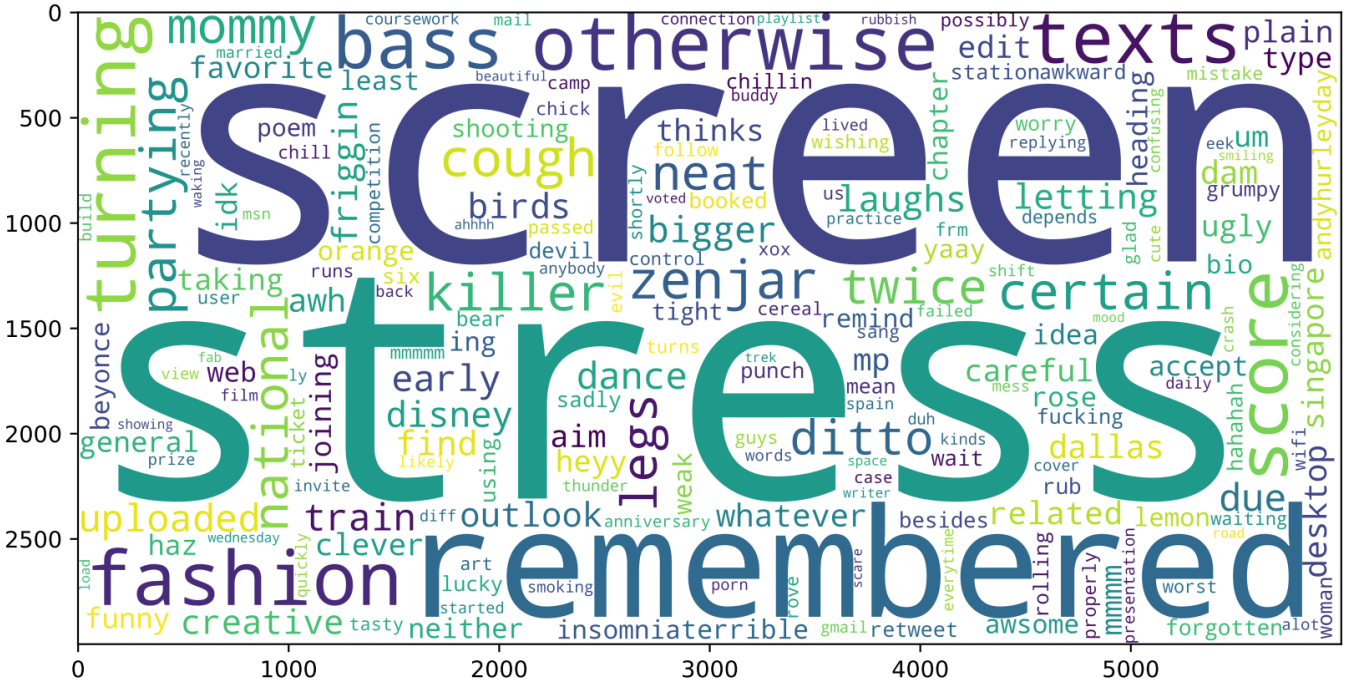
III. Results

We discovered that both Linear SVC and Logistic Regression perform best without the removal of stopwords;

[1]the value of *max_features* is not necessarily the same for both models



(a) *Positive tweets*



(b) *Negative tweets*

Fig. 4: Wordclouds derived from TF-IDF performed on tweets

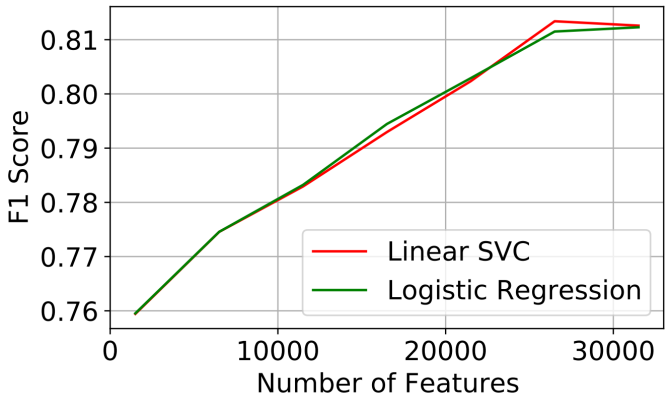


Fig. 5: Performance in terms of f1 score with increasing number of features

every following outcome has been obtained with the TF-IDF parameter stop_words='None'.

The results of the tuning of *max_features* are shown in Fig. 5. As expected, the score increases with the number of features considered. Both classifiers reach satisfactory results with *max_features* = 25000 and stabilize after. The best hyperparameter configuration for Linear SVC is {*C=1, dual=False, max_iter=2000, penalty=l1*} which granted us an *f1* score $\simeq$ 0.815, whereas the best configuration for Logistic Regression is {*'C': 5, 'dual': False, 'max_iter': 100, 'penalty': 'l2'*} with an *f1* score $\simeq$ 0.818. We train the best performing Linear SVC and Logistic Regression on the entire data available in the development set. We use the trained models to label the evaluation set. The public score obtained with the Linear SVC is 0.818 and the one obtained with the Logistic Regression is 0.821.

It is reasonable to assume there is no overfitting on our data because:

- the evaluation set has been used for the first time to compute the *f1* score seen on the leaderboard;
- the scores obtained with the 73/30 train/test split of the development set are almost identical to those in the leaderboard.

We can legitimately infer that the private scores will be similar to the public ones.

IV. Discussion

The proposed approach obtains results that outperform the baseline defined for this problem. It does so by applying an adequate preprocessing strategy and tuning hyperparameters of the considered classification models. The two selected classifiers perform similarly for this task and allow us to achieve performing results. Various aspects already discussed in this report may be worthy of further exploration:

- We have seen that removing stopwords from the text decreased the *f1* score on the Leaderboard. This behavior merits supplementary examination, in particular it could prove advantageous to define a custom list of stopwords for this specific dataset.
- Neural networks or deep learning techniques could considerably improve the score obtained with this methodology. Algorithms such as Doc2Vec or Word2Vec consider words' context and have been shown to achieve promising results in sentiment analysis [6].

The proposed classification problem is not trivial and there is room for vast improvement. Linear SVC and Logistic Regression, however, are two classification techniques that have been shown to obtain satisfactory results while maintaining a reasonable execution time.

References

[1] A. V. Aho, "Algorithms for finding patterns in strings, handbook of theoretical computer science (vol. a): algorithms and complexity," 1991.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[4] A. A. Effrosynidis D., Symeonidis S., "A comparison of pre-processing techniques for twitter sentiment analysis," *Research and Advanced Technology for Digital Libraries*.

[5] M. Saif, Hassan; Fernández, "On stopwords, filtering and data sparsity for sentiment analysis of twitter," *LREC 2014, Ninth International Conference on Language Resources and Evaluation Proceedings*.

[6] P. V. Maslova N., "Neural network doc2vec in automated sentiment analysis for short informal texts," *Speech and Computer. SPECOM 2017. Lecture Notes in Computer Science*.