An Explainable Approach to Twitter Sentiment Analysis

Francesco Capuano
Politecnico di Torino
Student id: s295366
s295366@studenti.polito.it

Alessia Leclercq
Politecnico di Torino
Student id: s291871
s291871@studenti.polito.it

***Abstract*—Sentiment Analysis is the task of identifying latent sentiment in text documents. This is a challenging assignment, particularly in the setting of user generated content, as it is necessary to analyze highly heterogeneous data. Here we present an explainable approach to this problem.**

I. Problem overview

The objective of this project was to train a model to perform automatic sentiment analysis and identify the latent sentiment of tweets, i.e., whether or not specific tweets convey a positive sentiment (indicated with 1) or a negative sentiment (indicated with 0). Sentiment Analysis is an analytical technique that extracts the writer's emotions and opinions from plain text, using traditional NLP rules or ML techniques. As observed in [1], Twitter represents a perfect sentiment analysis corpus because of its variety of users, social groups, and content. Hence, the heterogeneity of the micro blogging platform's is a valuable source for analysing people's opinions and feelings. The dataset was divided into two different sub-sections:

- A development set, containing 224994 labeled (*sentiment*) tweets with their content (*text*), some information about their publication (*ids*, *user*, *date*), and a query extraction field (*flag*). We have observed that this dataset does not present any missing values or duplicated records.
- An evaluation set, containing 74999 unlabeled tweets, with the same fields as the development set: sentiment prediction was performed on this set.

As previously observed, the data points appear to be extremely heterogeneous, hence a data exploration step was performed as the starting point for the classification task.

A. Feature Exploration

In this section we present the results of the exploration of the features.

a) user: It could not be assumed that the users in the evaluation set are at most a subset of the users in the development dataset. We focused on the distribution of the target variable among the users contained in the development dataset.

As shown in Fig. 1 we discovered that the majority of the users did not post tweets expressing predominantly one sentiment. Therefore, we concluded that the user column can be safely discarded as it was not a driver for the sentiment attribution.

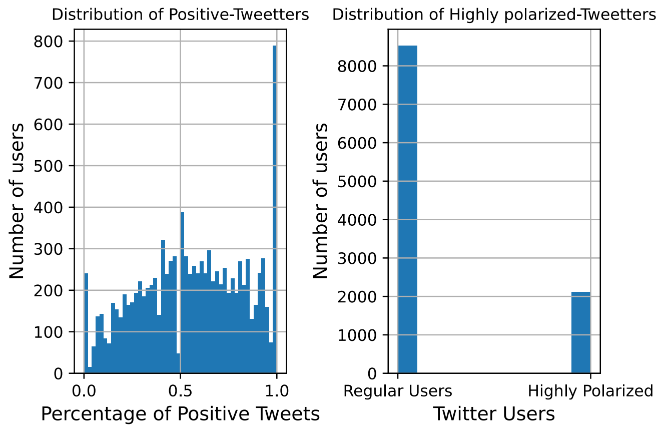

Fig. 1. Percentage of positive tweets on the total number of tweets per user (*left*). Distribution of the users with respect to the categories "Regular" and "Highly Polarized" (*right*)

b) flag: The flag column is related to the query used to extract the tweets. Since *NO_QUERY* was the only value, we decided to discard this column as well, as it was not possible to decouple the data points based on this attribute.

c) date and ids: The first contains information which related to each tweet's publication date, whereas the latter contains integers uniquely identifying each tweet. We discovered that the id is almost linearly correlated to the date.

Consequently, we could identify whether two tweets were consecutive in time simply by checking the ids. This seems a valuable discovery, as it introduces the notion of time-proximity between posts without dealing with posting time. However, since we believe that proximity between tweets does not convey any useful information for the sentiment analysis task, we decided to exclude both of those features.

d) text: The text column is the column in which the content of each tweet is contained. First, we observed that the tweets differed in terms of length (number of characters) and number of words.

As seen in Fig. 2, the vast majority of the tweets are within the 140-characters range allowed by the platform back in 2009, even though a small portion of them exceeds this limit, mainly because of noises in the tweets (e.g., unicode characters, emoji). This appeared to be a matter of data quality, considering the strict limit on the number of characters imposed in each tweet, hence we discarded those whose length exceeded the threshold. Moreover, Fig. 2, also includes the boxplot for the distribution of the length of each tweet. This second visualization shows that the values of length over 140 characters are statistically valid, even though they do represent a semantic error with respect to the context. This proves that merely limiting the feature analysis to statistical measures would have led to missing this crucial information related to the quality of our data.

We then focused on the number of words in each tweet. Again in this case, we found peculiarities in the data. Specifically, we noticed the presence of tweets composed of one single whitespace token. Considering the task of sentiment analysis, tweets with one word only are certainly not to be discarded as they may convey intrinsic, synthesized, sentiments. However, in the case of non-informative tokens (e.g., a mention, an URL, or a webpage), the single token tweets may be removed to improve data quality.

e) sentiment: Analysing the distribution of the target variable is crucial to understand whether or not the given classes are equally represented in the dataset. This distribution is shown in Table I.

Considering the result, we decided to undersample the ”positive” tweets, which made up the largest portion of the dataset. However, since the text cleaning steps that we discuss in the next sections were likely to affect the ratio of positive to negative, we decided to postpone the undersampling to the end of the text preprocessing phase.

	Absolute	**Percentage**
Positive	*130157*	*57,85%*
Negative	*94837*	*42,15%*
Total	224994	*100%*

TABLE I
DISTRIBUTION OF THE TARGET VARIABLE IN THE DATASET

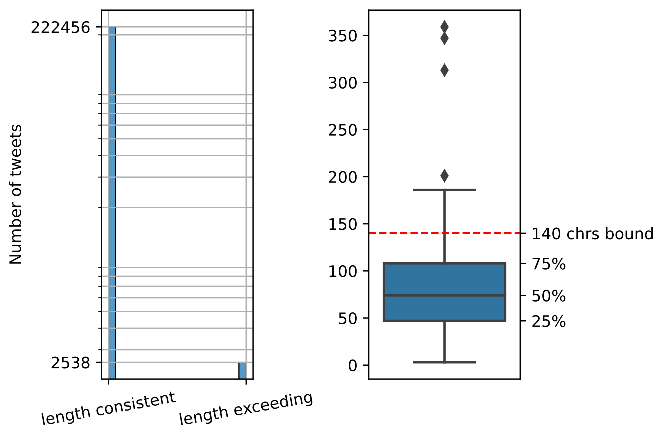

Fig. 2. Distribution of the tweets in the classes of those which exceed the 140 characters bound and those which do not (log scale, *left*). Boxplot for the number of characters of each tweet (*right*)

B. Feature Selection

According to the analysis just presented, we decided to discard all the original features except the *text* column.

II. Proposed approach

A. Preprocessing

Due to the informal nature of the tweets' content, many noisy aspects are encountered in the texts such as abbreviations, slang, emoticons, URLs, mentions, and hashtags. Considering the need to represent each tweet in a form that is intelligible to an ML model, we noticed that text cleaning was a necessary step before vectorization, both to ease the vectorization itself and to improve the quality of the representation.

a) Text Cleaning: The text cleaning process we designed for the Twitter texts is composed of the following different phases:

- *Removing Html tags*: We encountered Html tags (such as amp or reg) in the text. We cleaned the text by converting those to the actual textual element the tags represent.
- *Removing Twitter mentions*: Even if Twitter mentions represent the means of the exchange of tweets between users, we believe their presence would not add any information concerning our task (as shown in [2]), hence we decided to remove them.
- *Removing websites*: We have removed links to web resources from the tweets.
- *Expanding common contractions*: Considering the huge number of different kinds of users and the informality of the communications on the platform, contracted expressions repeatedly appeared in the texts. We considered the expansion of the contracted forms a way of reducing the noise inside the texts.
- *Removing non-letters characters*: We removed numbers, punctuation marks, and other characters.
- *Lowercasing*: We lowercased the whole string to reduce the dimensionality of the problem, as different-case versions of the same word are not necessarily proven to convey so different sentiments.
- *Replacing ”laughs” with <laugh>tag*: We believe the presence of laughs in the tweets is certainly a crucial feature for the given task. However, different versions (both in length and/or sequence of characters) of the normalized “ahahaha” word were found in the tweets. For this reason, we decided to replace every “haha”-like substring with a more general <laugh>tag.
- *Removing elongated words, multi-spaces, and single characters*: Considering the previous steps, different sequences of characters were replaced with white spaces, hence increasing the number of spaces that separate each word. Moreover, in the texts, different elongated versions of the same word are still present, as a consequence of the abovementioned informality of the platform. Considering that for the scope of sentiment analysis words like “waaaaayyy” and “waaayyyyyy” basically represent

Original Text	***Cleaned Text***
@Teri8D more &quot;bad things&quot; I've been there! It's WAAAAAAAAYYYYYY WORSE? http://blip.fm/∼7an09	more bad things have been there it is waaayyy worse
@Polito check www.polito.it	check
@vpsean ∼ Ahahahhha Thank you Sean hahahha	<laugh>thank you sean <laugh>
@SapienzaRoma check www.uniroma1.it	check
www.google.com	

TABLE II
EXAMPLES OF THE TEXT CLEANING PROCESS

the same thing, we decided to treat these elongated expressions reducing them to a more normal form. In the process of doing so, we also removed multiple whitespaces and words composed by one single character.

As shown in Table II, at the end of this process the original tweets assumed a cleaner and more standardized form.

The text cleaning process, however, introduced both empty texts and duplicates in the dataset. We justify this by noting that the reason for text cleaning itself was to reduce the heterogeneity of the dataset. This is why why we postponed the process of balancing the data to this very step : if we had done differently, we would almost inevitably have ended up in the situation of having to undersample the dataset twice, thus leading to an evitable reduction of the available data points.

Now, the just-produced cleaned version of the original text column had to be transformed into a format intelligible for ML techniques.

As a medium for the vectorization, we experimented with both the Tf-idf vectorization and the CountVectorizer transformation. Since different parameters of the two vectorizers are meant to be set when instantiating, we conduced experiments with different configurations of the two vectorizers. In particular, we compared these configurations with respect to the MacroAvg F1 score that we obtained by training a Logistic Regression Classifier on 95% of the data points, holding out a 5% portion for testing.

In particular, we have conducted experiments concerning:

- Whether or not to exclude stopwords.
- The maximum number of features.
- The value of n for the n-grams.

The results of our experiments are shown in Fig. 3.

We found that the best vectorizer for our task is the TfIdf vectorizer that:

- Does not exclude (predefined) stopwords set from the text (according to [3]).
- Represents the text with at most 300 thousand features.
- Uses unigrams, bigrams trigrams as features.

B. Model selection

We decided to test three models on the tfidf bigram vectorization using 95% for testing and 5% for the holdout. Considering the scant improvement consequent to increasing the value of *max_features* from 200k to 300k and adopting even tri-grams, for the sake of attempting to reduce the computational burden, we decided to set *max_features* equal to 200k and to stop at bigrams.

Namely, related to the models, we decided to test:

- *The naive Bayes classifier*: The choice of a Multinomial classifier rather than a Gaussian is due to the opportunity of using sparse data structures as inputs, resulting in a faster model fit and predictions computation.
- *The linear SVC*: Since we did not know the data distribution a priori and because of the high dimensionality of our problem, we decided to keep a linear model to trade off the computational time against the prediction accuracy. Even though the model might not fit our data the best (especially assuming a linear separation), SVC is between the best performing classifiers.
- *The Logistic Regression*: This model is interpretable since the higher coefficients (in absolute terms) correspond to the most relevant words in terms of sentiment attribution.

The models' results are present in Table III.

Classifier	***MacroAvg F1 Score***	***ROC-AUC score***
Logistic Regression	0.80	0.8869
Linear SVC	0.79	0.8756
Naive Bayes	0.79	0.8809

TABLE III
MODELS COMPARISON

Even though the difference in performance is only 0.01, we chose the Logistic Regression model to perform hyperparameter tuning and to do the predictions. This choice was driven by higher performance and interpretability since we can identify the most relevant words for positive and negative sentiment prediction only by checking the values of the model-estimated coefficients, thus leading to having an explainable model.

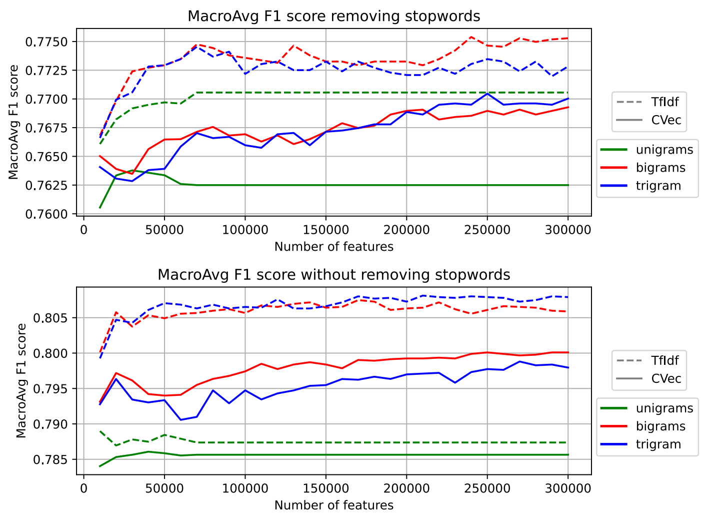

Fig. 3. MacroAvg F1 score obtained with Logistic Regression with different vectorizers

hyperparamether	***value***
class_weight	*None*
fit_intercept	*True*
intercept_scaling	*1*
l1_ratio	*None*
max_iter	*100*
multi_class	*auto*
penalty	*l2*
solver	*lbfgs*
tol	$1e^{-4}$
C	*1.7237*

TABLE IV
RESULTS OF THE GRID SEARCH

C. Hyperparameters tuning

We performed a grid search on the best-performing model, the logistic regression, and the dataset preprocessed using uni-grams and bi-grams. The results of the grid search are presented in Table IV. In particular, we have experimented using cross-validation with five folds.

III. Results

We have presented a Logistic Regression classifier with the parameters presented in Table IV. This particular model reached a value of MacroAvg F1 score of 0.8048 in the eight folds cross validation we used to tune the hyperparamethers and a value of 0.802 in the leaderboard. The ROC curve for this classifier is presented in Fig. 4

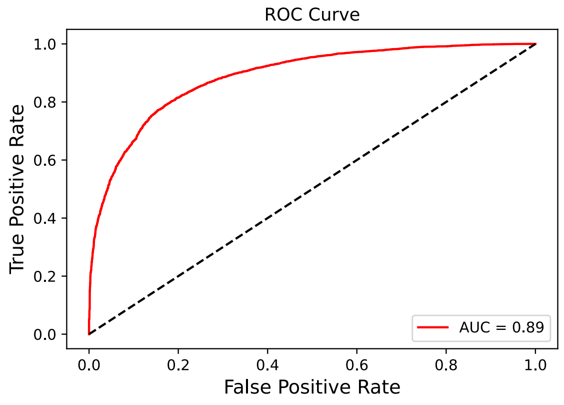

Fig. 4. ROC curve with AUC score for our best classifier

IV. Discussion

As presented in [3], removing predetermined standard set of stopwords in the text feature extraction step typically affects negatively the performance. As this work has shown, stopwords removal should be considered only when scope-tailored sets are available. Considering the high heterogeneity of the context, we have not defined a custom set, even though having one would definitely have helped out.

As we have previously highlighted, a linear model such as Logistic Regression has been used considering the possibility of giving explanations for this specific model. We have investigated the weights the model have learned for the top 5 words in terms of positive-sentiment attribution and negative-sentiment attribution. The results are shown in Table V.

Top positive words	**Top negative words**
cannot wait	*sad*
not bad	*poor*
awesome	*miss*
yay	*cannot*
thanks	*hate*

TABLE V
MOST INFLUENT WORDS, SORTED WITH RESPECT TO THE WEIGHTS LEARNED BY THE MODEL

As these results show, the model has learned the latent sentiment conveyed by n-grams such as *cannot wait* or *poor*. It is interesting to note how strongly positive sentiment appears to be connected to words that typically express a sense of excitement for the upcoming such as *"cannot wait"*. We conclude this work by noting, on the contrary, how negative emotions tent appear be related to the feeling of impotence (e.g., *"cannot"*).

References

[1] A. Pak and P. Paroubek, “Twitter as a corpus for sentiment analysis and opinion mining.,” in *LREc*, vol. 10, pp. 1320–1326, 2010.
[2] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, “A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis,” *Expert Systems with Applications*, vol. 110, pp. 298–310, 2018.
[3] H. Saif, M. Fernández, Y. He, and H. Alani, “On stopwords, filtering and data sparsity for sentiment analysis of twitter,” 2014.