Predicting Tweets Sentiments With Machine Learning Algorithms

Arcangelo Frigiola
Politecnico di Torino
arcangelo.frigiola@studenti.polito.it
s295406

Gianluca La Malfa
Politecnico di Torino
gianluca.lamalfa@studenti.polito.it
s290187

***Abstract*—In this report we introduce a possible approach to the Tweets Sentiment Classification problem. In particular the proposed approach consists of choosing which are the most influential features of the dataset and build a classification pipeline. The proposed approach outperforms a naive baseline defined for the problem and obtains overall satisfactory results.**

I. Problem Overview

The goal of the competition is analyzing and predicting which is the sentiment of each tweet. We are working on a dataset that has been divided into two sections:

- *Development set*: it is composed by 224994 records
- *Evaluation set*: it is composed by 74999 records

A deep study of the development set will make us able to understand how the data is generally distributed and above all to build a model that will lead us to predict the sentiments of the sentences in the evaluation set. The attributes that compose our data are:

1) *Ids*: a numerical identifier of the tweet;
2) *Date*: the publication date of the tweet;
3) *Flag*: the query used to collect the tweet;
4) *User*: the username of the original poster of the tweet;
5) *Text*: the text of the tweet;
6) *Sentiment*: the sentiment of the tweet, our target variable. This attribute can only assume two values:
   0: if the tweet express negative feelings;
   1: if the tweet express positive feelings.

TABLE I
Dataset composition before any preprocess step

	Number of unique values	Number of missing values
ids	224716	0
date	189779	0
flag	1	0
user	10647	0
text	223106	0

After a first look at the dataset, our first concern was to check the eventual presence of duplicates. This process is very important because it can prevent us from obtaining misleading statistics. 1888 records containing duplicate text have been found, so we decided to drop them, keeping only one copy of each. Next, we started looking for null values, although we did not find any. Finally we managed the *date* attribute, splitting it into all of its components: years, months, days, hours, minutes and seconds.

The dataset on which we are working on seems to be slightly unbalanced. As we can see from Fig.1, there is an higher percentage of text labeled as positive with respect to the ones labeled as negative. This could slightly influence our process pipeline, because our machine learning algorithm could be biased towards the majority class. However, the gap is not so relevant to compromise the classification.

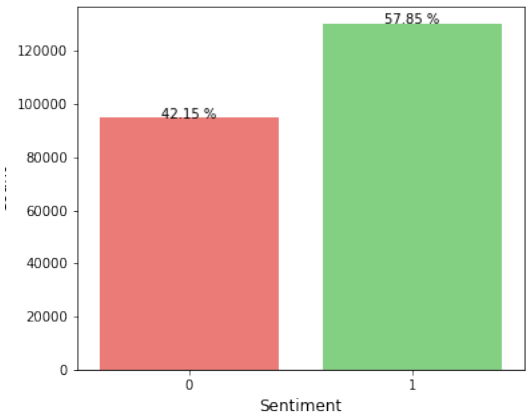

Fig. 1. Number of Negative and Positive Tweets

II. Proposed Approach

A. Preprocessing

After managing the missing and duplicates values, we focused our attention on deeply understanding what the composition of the dataset was. As concerns the attribute *ids* it represents only the unique identifier of the tweet and does not provide any useful information related to the purpose of our studies so we decided to not consider it. As regards the *flag* attribute, as we can see from the Table I we have just one unique value which is `NO_QUERY`, this means that no additional information is provided by it, so we decided to drop it as well. Considering this, we worked only with *user*, *date* and *text*, because of their correlation with the target value.

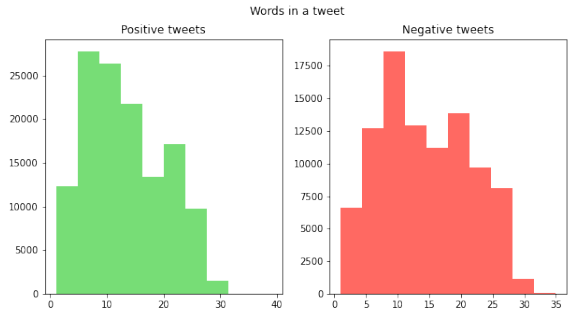

Fig. 2. Number of words in a tweet

We decided to work with *user*, because we considered that people tends to have a behaviour which is approximately uniform over time, so it could be a good indicator for our purpose. While the reasons why we kept all of the components of *date* was that, when tweeting people tends to talk about their stuff or comment on public events. As a consequence we assumed that what is happening around the word can influence the tweets made during those times. The only attributes related to *date* we dropped were *year* because all the tweets are from 2009, so no additional information was included, *minutes* and *seconds* because of the excessive precision not adequate to the faced problem.

The next and core section of our study was the analysis and cleaning of the text of the tweets. This step is a crucial one and will strongly influence the outcome of our model predictions. First we applied some basic cleaning preprocessing steps: we transformed all the text in lower case in order to reduce all the possible differences between the words with the same meaning but with different writings, we removed all the links, the punctuation, considering also "@" and "#" which are very common symbols in tweets, emojis and non ASCII characters. Finally we also removed the stopwords. More precisely we created a list containing only the most frequent ones in order to remove redundancy while keeping as much information as possible. Moreover stemming was not applied, despite it is known that theoretically the application of this technique would result in a reduction of noisy data. Indeed, in this scenario we would have lost a lot of important information about the real sentiment of the word resulting in a lower efficiency of our pipeline.

At this point, one of the main problems we had to face was represented by the presence of abbreviations in our sentences. It must be considered that these tweets are used by users to express their feelings or simply communicate with their friends. This results in a text which contains a lot of expressions from the spoken language and abbreviations, even due to a maximum number of words available. So our idea was to create a dictionary containing all the most frequent "slangs" and abbreviations in order to translate them.

After these steps we focused on the main differences between the tweets which expresses positive feelings and negative ones. We separated them and we studied their composition. After our analysis, we found that positive tweets tend to have more words in their sentences, as can be seen in Fig.2. We also decided to study the average word length in each tweet, the results are reported in Fig.3. Since there are not significant differences, we did not go into further details.

After this, we tried to understand which are the most common words in both positive and negative sentence, in order to understand which are the most influential, as it is possible to see in Fig.4 and Fig.5.

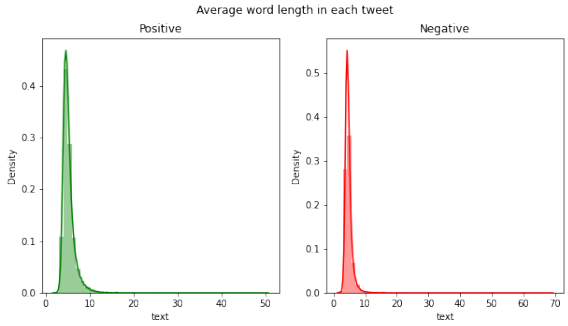

Fig. 3. Average word length in each tweet

As we know, it is impossible to feed our machine learning algorithm with non-numerical features, so before going on, it is necessary to perform an encoding phase. The encoding techniques we explored are:

- One-hot Encoding for *users* and all components of *date*: One-hot encoding is used in machine learning as a method to quantify categorical data. This method produces a vector with length equal to the number of categories in the data set. If a data point belongs to the *i*th category, then components of this vector are assigned the value 0, except for the *i*th component, which is assigned a value of 1. In this way it is possible to keep track of the categories in a numerically meaningful way.
- TF-IDF (term frequency-inverse document frequency) for *text*: it is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document rank low even though they may appear many times, since they do not mean much to that document in particular. In our case, we set the number of n-grams to three, trying to save as much information as possible from the text.

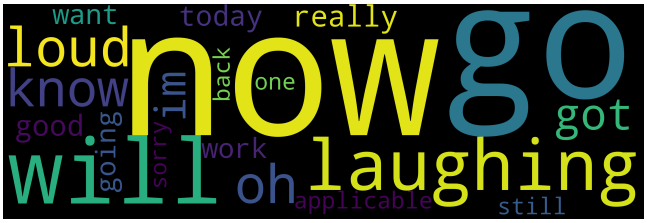

Fig. 4. Top 25 words in negative tweets

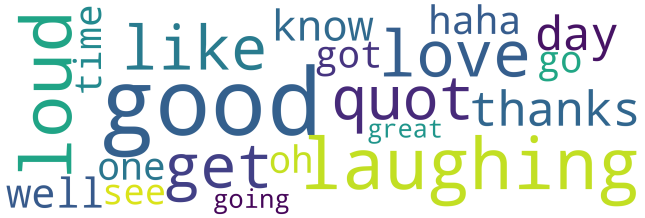

Fig. 5. Top 25 words in positive tweets

B. Model selection

Considering the nature of the problem, and the composition of the dataset we are working on, we considered the algorithms that could theoretically perform better and tested them.

- *Logistic Regression*: it is a simple regressor model which fits perfectly for binary classification problems (e.g. Yes or No, 0 or 1). It uses a method known as maximum likelihood estimation to find the best fitting regression equation.
- *Linear SVC*: it is a Support Vector Machine - SVM. As supervised learning methods, they solve the optimization problem of finding the hyperplane that maximizes the margin between two classes in the feature space. Furthermore, it is among the best performers in this kind of tasks, and in robustness to noise and outliers.
- *Naïve Bayes Classifier*: it is the simplest and fastest classification algorithm for a large chunk of data. Based on the Bayes theorem, there are essentially three types of NBClassifiers: Gaussian, Multinomial, and Bernoulli. For the *Sentiment Analysis Problem*, we decided to adopt the Multinomial one, which is useful to model feature vectors where each value represents, for example, the number of occurrences of a term or its relative frequency; in this case we refer to the TF-IDF output.

We decided not to try polynomial solutions due to the high cardinality of the problem under analysis.

In the following section, we will discuss the parameters we used and their values in order to build the grid search for the hyperparameter tuning.

C. Hyperparameters tuning

In order to find the best parameters, we ran a grid search for each model chosen, as shown in Table II, evaluating it through F1 macro index.

TABLE II
HYPERPARANETERS TABLE

Model	Parameters	Values
Logistic Regression	penalty	'l2','**'none'**
	tol	1e-1, **1e-2**, 1e-3, 1e-4
	C	**0.6**, 0.8, 1.0, 1.2, 1.4
	solver	'saga'
Linear SVC	penalty	'l1','**l2**','none'
	loss	'hinge','**squared_hinge**'
	tol	**1e-1**, 1e-2, 1e-3, 1e-4
	C	0.1,0.2,**0.3**,0.4,0.5,0.6,0.8,1.0
Naïve Bayes Classifier	alpha :	**0.2**,0.4,0.6,0.8,1

III. RESULTS

From the *hyperparameters tuning* phase it has emerged that the best performance was achieved with the linear svc model with a final local score of 0.856, with the following parameter-value matches:

- *C - 0.3*,
- *loss - squared_hinge*,
- *penalty - l2*,
- *tol - 0.1*.

For the sake of completeness, it is possible to see the best combination of parameters for all the models highlighted in Table II.

We trained the best performing model on all available development data. Then the model has been used for the evaluation set. The public score obtained is 0.853. Since this score is obtained using the evaluation data, there should be no overfitting and we can reasonably assume that similar results can be obtained on the private score.

IV. DISCUSSION

The model obtained performs very well, going beyond the 0.753 threshold posed on the leaderboard. This result can be interpreted as a good processing of all the steps of the *Sentiment Analysis Problem*.

Anyway, even though we achieved a noticeable score, we are conscious that an even more accurate model can be built. *Sentiment Analysis* is one of the most requested competences in Data Science and Data Analysis fields, and more precise and powerful tools have been developed ad-hoc for Natural Language Processing problems and text analysis (e.g. ELMo, BERT, and transformers in general).

Furthermore, a deeper approach can be done in the pre-processing phase, making the dataset completely balanced. As remarked in previous sections, the *Negative* and *Positive* tweets are slightly unbalanced - 42.1% and 57.9% respectively. To deal with this issue is possible to use two mathematical techniques: *Over Sampling* and *Under Sampling*. To conclude, we are satisfied of the results achieved, but aware that higher scores could be reached.