Tweet's sentiment classification

Chiara Van der Putten and Pietro D'orto
Politecnico di Torino
Student id: s264255 and s297340
s264255@studenti.polito.it and s297340@studenti.polito.it

***Abstract*—In this report we develop a possible solution for a problem of predicting the sentiment of a tweet by means of classification techniques. We decided to focus a lot on data pre-processing and on the extraction of the most important feature engineering that characterize the sentiment of a tweet. The solution compares different classification models and the results obtained with them.**

I. Problem overview

The following project is a classification problem about tweet's sentiment. The target of the project is to identify the sentiment of a tweet which can be '1' if it is positive or '0' if it is negative.
We will use DSL2122_january_dataset dataset, it is composed by two files:

- development.csv composed of 224994 rows characterized by the numerical target, a tweet text and a set of categorical features, containing some information about the tweet (date, user, ids, flag)
- evaluation.csv composed of 74999 rows with the same structure of the development set, except for the numerical target.

We first looked for missing or duplicate values to handle them properly. From this research we noticed that there are no missing values while duplicates are about 278, duplicate tweets do not bring additional information so we removed them, these were found through the 'ids' field, which uniquely identifies a tweet.
Now we can start analyzing our data and their correlations we may exploit in further phases.
Starting with the distribution of sentiment we can see that the number of positive and negative tweets is quite balanced as shown in Fig.1 . In fact, the negative tweets are 94837 while the positive ones are 130157, so we can proceed with further analyzes on the other fields.
In Fig.2 it is shown the correlation between users and the sentiment of the tweets they write. Each user in the graph is defined by the pairs (#positive_tweet , #negative_tweet). As we can see there is a strong correlation between the sentiment of a tweet and who writes it, so we decided to take that into account to building our model, how we do this is better described in the paragraph dedicated to the pre-processing. Another correlation we decided to explore is the one between the days of the week and the sentiment's distribution of tweets on those days (Fig.3). The correlation is not as strong as the previous one

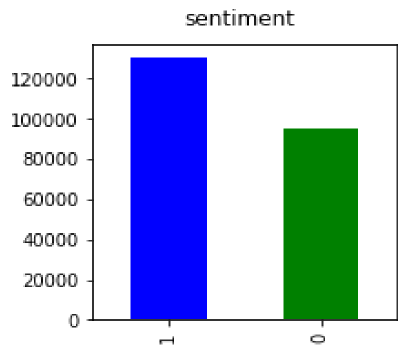

Fig. 1: Distribution of development set sentiment

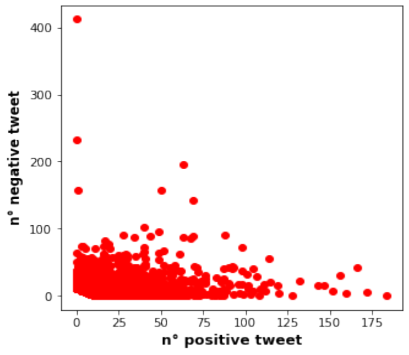

Fig. 2: Number of positive and negative tweets per user

but there are days like Monday and Sunday in which there is a prevalence of positive tweets, so we decided to take into account as well.

After these first considerations we noticed that all the lines in the flag field were set to NO_QUERY so we decided to eliminate the flag column as it is irrelevant to discriminate the sentiment of the tweet.
A disturbing factor for the analysis could be the web addresses that could be replaced as they do not constitute an important element for our analysis.Even stop words, like: 'e' 'also' 'that' 'some', slow down the classification process and therefore need to be removed.

II. Proposed approach

A. Preprocessing

The data extracted online, especially from social networks, is often unstructured and may contain unnecessary data. Indeed a large number of features needs more time for training and

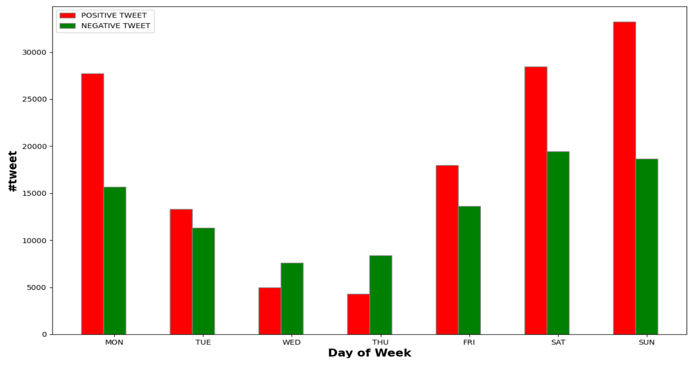

Fig. 3: Sentiment distribution of tweets over the days of the week

stop words, punctuation marks, URL, numerical values and data that are not related to the analysis reduce the accuracy of the forecast. Pre-processing is really important to economize computing resources and makes it easier for models to train more effectively, giving a precise prediction. So we applied a lot of steps in the pre-processing part:
First we turned the tweet text to lowercase, in this way we do not have duplicate identical words. In addition to this we have also eliminated all the punctuation, numbers, user tag and URLs that could cause a waste of energy for the model given their variety.
After that we transformed the words into their root forms by removing the affixes using the lemmatization and stemming.They also transform plurals into singulars and contracted forms into the non-abbreviated form.
They also eliminate stopwords to which we have also added words such as 'quote' and 'amp' that we noticed in the tweets due to a wrong translation of punctuation from HTML.
We have also added words such as "im", "go" "day", "one" to the stopwords set, so we can eliminate them, as they were present very frequently in both positive and negative tweets and therefore they are not capable to discriminate a positive tweet from a negative one and vice versa.
Figures 4 and 5 show us which are the most recurrent words in positive and negative tweets after the cleaning step.
Most data mining algorithms are unable to directly process textual data in their original form so these must be transformed into a more manageable representation like a feature vector. To do this we used TfidfVectorizer from sklearn, that assign to each term a value which represent his frequency in a collection of documents according to this formula:

$$\text{(For.1) tf-idf}(t) = \text{freq}(t,d) * \log(m/\text{freq}(t,D))$$

where:

- t represent the term;
- d represent the document;
- D (consisting of m documents) represent the collection of document.

In this way we are able to emphasize those terms that occur frequently in a single document but rarely in the whole collection.
As mentioned in the Problem Overview's chapter, the features we have taken into account are: tweet's text, day of the week and user.
First we checked out manually the day of the week from the date feature and created a new column containing them, this was done both for development and evaluation data sets.
After that we used three different TfidfVectorizer, one for each feature, and then combine them, using hstack method from spicy.sparse package, to be able to pass them to the models. In this way we composed a set of features composed by a set of 1057204 word's features, 7 days of week and 10647 users, all of them are weighted according to (For.1).
We tried different combinations of values for the parameters of TfidfVectorizer using a simple Pipeline to evaluate which one gave us the best values in terms of memory used, computation time and score. For the words we have decided to keep only the parameter ngram_range, that we have set to (1,2) which is equivalent to considering Bigrams, while for users and days of the week we've decided not to use any parameter considering therefore all the features. This is because we would like our model to be able to predict a wide range of features.

B. *Model selection*

The models we have decided to test are the following:

- Logistic Regression: Produces the association between the variable, to be predicted, and one or more independent variables by approximating the probabilities using a sigmoid function [1]. We decided to test the LR as it gives better results when the target class is categorical.
- Random Forest Classifier: It combines multiple decision trees to make predictions. The model trains on different portions of the training set to increase performance and to avoid overfitting. We have chosen to use this model because it is suitable for handling large size noisy data in text classification, furthermore it is very expensive in terms of training time.
- LinearSVC: The model applies a linear kernel function to perform the classification and works well with a large number of samples, which is why we decided to test it on our data. We also used this model because, according to the sklearn documentation, it should scale better to large numbers of samples.

To have a starting point of reference, we first tried the various model with a simplified version of the features that is the only text.
We noticed that using only the text of the tweets to predict the sentiment, the result we obtained for F1 remained around 0.77. We therefore decided to include the day of the week in the meaningful features, thereby increasing the score to around

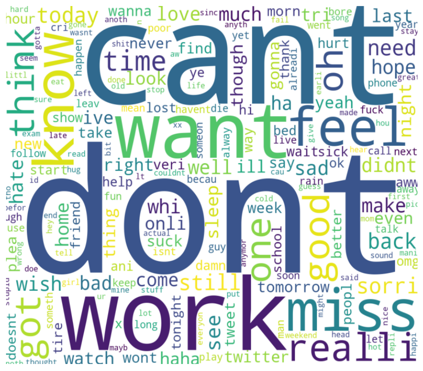

Fig. 5: WordCloud of negative tweets

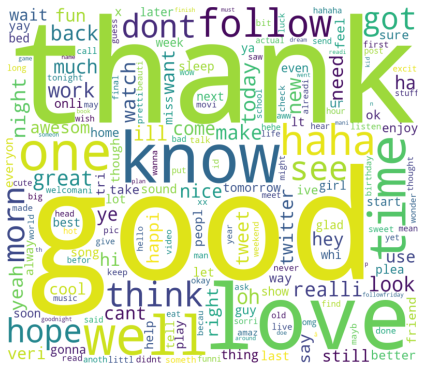

Fig. 4: WordCloud of positive tweets

0.79. Finally, we also included users in the analysis, the latter figure significantly improved the prediction of the sentiment of tweets, bringing the result to 0.82. So all the analyzes that we will see from here on have been done on the best turned out dataset to be text, day of week and user.
For all models, the best configuration of the hyperparameters was determined using a grid search, as we will see in the next paragraph.

C. Hyperparameters tuning

For tuning our models we first need to split the development data set, because of its size. We decide to keep 80% of the observations in the training set, and use the remaining 20% for the test set. We used the first to train the model

TABLE I
GRID SEARCH PARAMETERS

Model	**Parameter**	**Values**
Logistic Regression	*solver*	liblinear
	C	0.5, 5, 10
	random_state	42
	multi_class	ovr
	max_iter	10, 30, 300, 2000
Random Forest Classifier	*n_estimators*	50, 250, 500
	max_features	sqrt , log2
	max_depth	50, 500, 1000
LinearSVC	*C*	0.05, 0.6, 5
	multi_class	ovr
	random_state	42

TABLE II
TABLE II: HYPERPARAMETERS TUNING FOR LOGISTIC REGRESSION

C	max_iter	F1 Score
0.5	30	0.808
0.5	300	0.804
5	30	0.816
5	300	0.81
10	30	0.81
10	2000	0.80
100	30	0.81
100	2000	0.80

configurations, the second to evaluate them and then choose the best model and its configuration.
Then we used GridSearch which tests different combinations of parameters in the different models and finds for each the combination with the best F1_macro score.The values we have decided to test are shown in Table I.

III. RESULTS

After selecting these parameters for the templates, we ran the grid search for the classifiers and we identified the following best configurations:

From the Table II we can see the best configuration for Logistic Regression is: { solver=liblinear, C=5, random_state=42, max_iter=30 } (F1 Score $\approx$ 0.816), but we can also note that the increase in the value of F1 is strongly linked to max_iter value, in fact we have seen that the lower its value the better the result as shown in Table II. Anyway you have to be careful to reduce the value of this parameter because otherwise the model is unable to converge.

The Random Forest Classifier achieved the best results with the following configuration: { n_estiamtor= 500, max_features= sqrt, max_depth= 500 } (F1 Score $\approx$ 0.76). By observing Table IV we can see how higher values correspond to a higher max_depth, the same also happens for the number of estimators, as the number of estimators increases, a higher F1 Score is obtained. It can also be observed how the log2 value of max_features significantly lowers the result returned to us by the model. We also want to say that this model was the only one in which we had to reduce the number of features otherwise the computation time would have been too high.

For LinearSVC the best combination of parameters is: { C=0.6, multi_class=ovr, random_state=42 } (F1 Score $\approx$ 0.82). We can see that the model returns the highest score for F1 with low values of C.

The three classifier models show similar results but we can identify LinearSVC as the best performer followed by Logistic Regression and Random Forest Classifier. After obtaining these results locally, we used the entire development set to

TABLE III
TABLE II: HYPERPARAMETERS TUNING FOR LINEARSVC

C	F1 Score
0.05	0.804
0.6	0.82
5	0.80

TABLE IV
TABLE IV: HYPERPARAMETERS TUNING FOR RANDOM FOREST CLASSIFIER

Max_Depth	N_Estimators	Max_Feature	F1 Score
50	50	log2	0.39
50	250	log2	0.37
50	500	sqrt	0.68
500	50	sqrt	0.75
500	250	log2	0.74
500	500	sqrt	0.76
1000	50	sqrt	0.75
1000	250	sqrt	0.757
1000	500	sqrt	0.75

train the different classifier models with the best combination of parameters. Then the algorithms were applied to the evaluation set to predict the sentiment of the tweets and the scores we obtained are the following: 0.824 for LinearSVC.

IV. DISCUSSION

The difference between the score obtained in the leaderboard and that obtained privately is quite small. The score obtained by the best model is 0.82 while the one submitted publicly is 0.824.
We can visualize the performance of our model by means the AUC-ROC curve (Fig.6) and the confusion matrix (Fig.7).
A higher True Posn and a lower False Negn is desirable since we want to correctly classify.We can see that the value of true negatives is less than that of true positives, this could be due to the greater presence of positive tweets compared to negatives in the development set. In fact, in all the models used we noticed an increasingly higher F1 Score in positive tweets. The AUC-ROC curve tells us how much our model is capable of distinguish between classes.
In our case the score could have been improved by trying more parameter (like days of week divided by each month or each day of each month) combinations for the various models. Unfortunately, this would add a lot of features to train the models, that is already made up with all users, words and days of the week, and the computational training time would become too long.
There are also some parameters in the pre-processing phase that could have improved the analysis. For example, the hashtags that we have considered as normal words could have been valued more considering their importance in emphasizing the sentiment of the tweet, but also the use of tools like TextBlob [2] to identify and possibly remove tweets with a more neutral sentiment that do not help in discrimination.

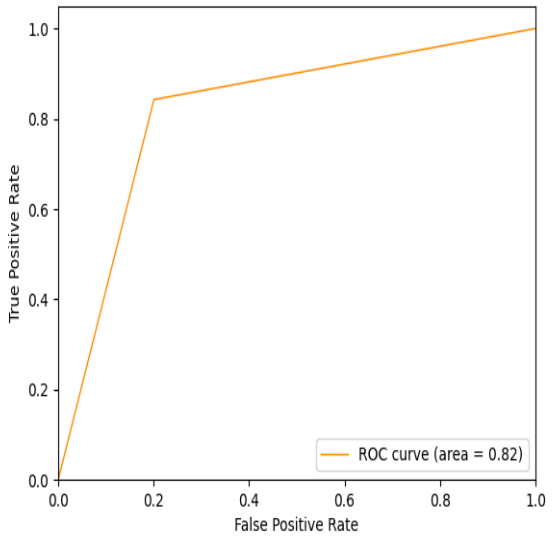

Fig. 6: AUC-ROC LinearSVC curve

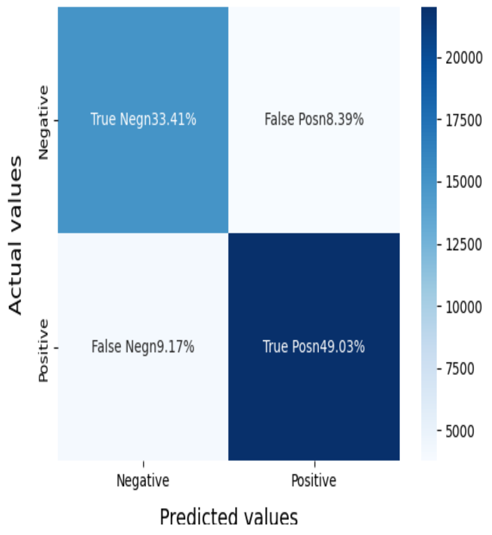

Fig. 7: Confusion matrix

V. REFERENCES

[1] Babacar Gaye, Dezheng Zhang and Aziguli Wulamu, ”A Tweet Sentiment Classification Approach Using a Hybrid Stacked Ensemble Technique”, University of Science and Technology Beijing.

[2] Sentiment Analysis using TextBlob https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524