Politecnico di Torino

01TWZSM Data science lab: process and methods

A.A. 2019-2020

July session exam

Project Report

Student:

Valerio Zingarelli (234275)

Professor:

Prof. Elena Maria Baralis

Assisants:

Tania Cerquitelli
Andrea Pasini
Giuseppe Attanasio
Flavio Giobergia

Contents

1 Data Exploration

The dataset we are required to analyze is a collection of posts from Twitter social network regarding BlackLivesMatter movement. The task of the project is to make a sentiment analysis of these posts. All the tweets are written in the English language and contain only plain text and at least one keyword connected to the movement. After using pandas to read and store the dataset in a proper data structure, I started to explore it in order to understand the way it's composed. Each line of the file is characterized by multiple attributes:

- created_at: the date of creation of the tweet
- id : the unique identifier of the tweet
- full_text: the tweet itself
- user: the user who wrote the post
- retweet_count: the number of retweets
- favourite_count: how many times the tweet has been liked
- coordinates: it represents the location of the Tweet
- place: it indicates if the tweet is associated with a place
- class_label: it can get 1 if the tweet shows a positive sentiment about the movement or 0, otherwise.

Class labels are surprisingly balanced: I noticed that 49.9% of them are positive posts and 50.1% are negative as it's possible to understand from the bar chart below:

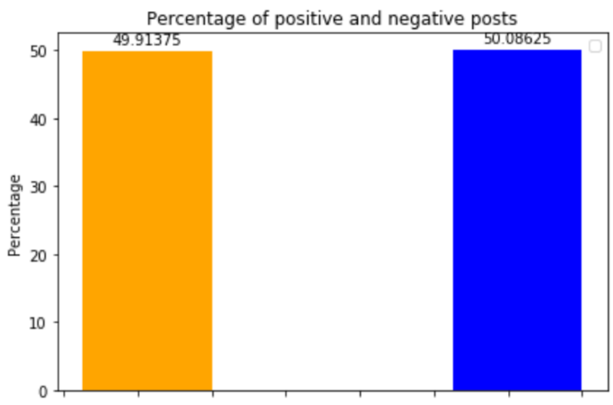

Figure 1: Percentage of Positive(left) and Negative(right) tweets

After that, I decided to use a useful tool to deal with text: WordCloud. It's a visual representation of textual data where the bigger is the word the more frequent it is. This is the result obtained using the previous tool on the dataset containing the tweets.

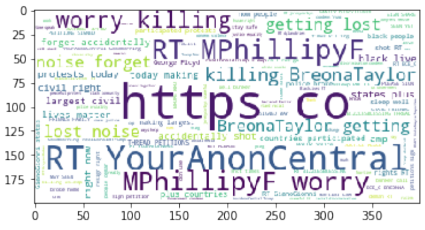

Figure 2: Positive posts WordCloud

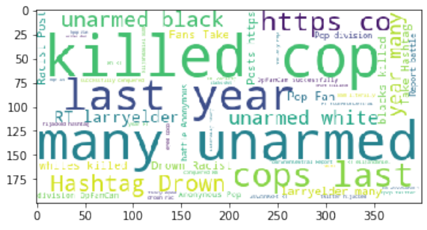

Figure 3: Negative posts WordCloud

It's possible to notice that there are different words used in both classes and that there are "useless" words , such as "https", that will be deleted during the processing.

2 Pre-processing

For what concerns the Pre-Processing part, I decided to use only the "full_text" column of the dataset and to apply Tokenization to it.

2.1 Tokenization

Tokenization is a technique allowing us to split sentences into smaller pieces in order to extract only the central and most important part of them.

2.2 Stopwords

Before applying the TfIdfVectorizer tool, I removed stopwords: these are words like prepositions or articles that do not help us to recognize if a document expresses a positive or negative sentiment and, so, they are useless for our purpose. I used the English default stopwords and I added "https" and abbreviations like "u" or "ha" to them.

2.3 Tf-Idf Vectorizer

After having tokenized the text, I applied TfIdfVectorizer which creates a matrix with all the words and their scores in all the documents. This is a really performing approach when we have to deal with heterogeneous documents. Furthermore, using this scheme, terms occurring frequently in a document but rarely in the others are preferred.

Then, I proceeded with the choice of the Algorithm.

3 Algorithm Choice

The first algorithm I tried was the DecisionTree Classifier, followed by its natural evolution: RandomForest Classifier. With both of them, changing all the parameters in different combinations , I got a maximum f1 score of more or less 0.92.
So I decided to look for something else.
Since we have 2 classes, we are looking for something capable of dividing our points into two sets in order to classify them properly. I found out that the best way to obtain such a result is to use SVC or, more precisely, a variation of the classical SVC called LinearSVC which is more suitable for large scale problems because of its velocity which is higher than the classical SVC's one.
With a C value of 4, I got the best result: 0.9357 in the Public Leaderboard.
After trying LinearSVC Classifier, I also tried to use other classifiers ,like Bernoulli NaiveBayes or MultinomialNB, but I got 0.9283 as maximum score so I decided to keep using LinearSVC.

4 Tuning and Validation

In order to validate my model I partitioned data into a training set and a test set using train_test_split from SKlearn with test_size=0.2 in order to evaluate locally my model before trying to commit the result file. To evaluate the model's performances, the selected metrics is F1 score which is an harmonic mean between Precision and Recall.

$$F1 = 2\,\frac{p \cdot r}{p + r} \tag{1}$$

In order to maximize my f1_score, I used the following settings in the LinearSVC model:

- tol= 1e-10 : the tolerance of the algorithm
- C=4 : the regularization parameter
- loss= "hinge": the standard SVM loss
- dual= True
- max_iter= 100000: the maximum number of iterations

As I wrote before, in the pre-processing part I used the TfIdfVectorizer algorithm. The parameters I set are the following:

- ngram_range=(1,4): it splits sentences into tokens of dimension 1,2,3 or 4.
- stop_words: I used the english default stopwords and I added some others stopwords like abbreviations.
- use_idf= True: it enables the inverse-document-frequency reweighting

All the other settings not appearing in this list were set with the default values because I did not notice improvements deriving from their modification.