Sentiment prediction of Twitter contents

Edoardo Chiò
Politecnico di Torino
Student id: s301486
s301486@studenti.polito.it

***Abstract*—In this report a sentiment analysis of a collection of Twitter posts is addressed. The proposed approach starts with several text preprocessing steps, that include dataset cleaning and word stemming, and lastly the creation of a bag-of-words weighted using the tf-idf method. The obtained transformed dataset is then used to train and validate two models, that are a Linear Support Vector Machine Classifier and a Multinomial Naive Bayes Classifier. The tuned models, using the f1 macro score as evaluation metric, are able to outperform the baseline and to reach good results.**

I. Problem overview

This project concerns a classification problem applied to a collection of Twitter posts (i.e., *tweets*) written by different users. The goal of the project is to perform a sentiment analysis of the posts contained in the dataset.
Training and validation of the models are conducted on a development set, containing 224,994 labelled recordings, while the set to test the models contains 74,999 recordings.
The development set is composed of six fields:

- *sentiment*: sentiment labels;
- *ids*: numerical identifier of the tweet;
- *date*: publication date of the tweet;
- *flag*: query used to collect the tweet;
- *user*: name of the user that posted the tweet;
- *text*: text of the tweet.

The *sentiment* field contains a label for each record. There are two classes: the text is considered having a positive trait if the label value is **1**, instead it is considered negative if the label value is **0**. The classes are not well balanced, indeed there are 130,157 data points having label 1, and just 94,837 having value 0. An exploratory analysis of the dataset was performed, to study the contents of all the fields and understand which features should be taken into account.
The *ids* field (i.e., the row identifiers) presents 278 pairs of duplicates, each one having one row of the related pair of rows associated to a positive sentiment, and the other one to a negative sentiment; since it is not possible to know the correct sentiment for these records, they do not add any insight and so they were removed during the data cleaning.
The *date* field shows that the tweets, both in the development dataset and in the evaluation dataset, were posted between April 6th 2009 and June 25th 2009; this suggests that the data should be homogeneous across the datasets, with a similar distribution of sentiment.
The *flag* field presents a unique value in all tweets in both the datasets, so it is not useful for the analysis and it can be discarded.

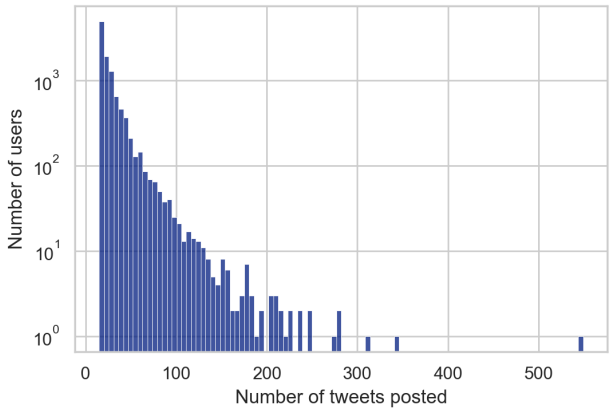

Fig. 1. Histogram representing the number of users per number of tweets

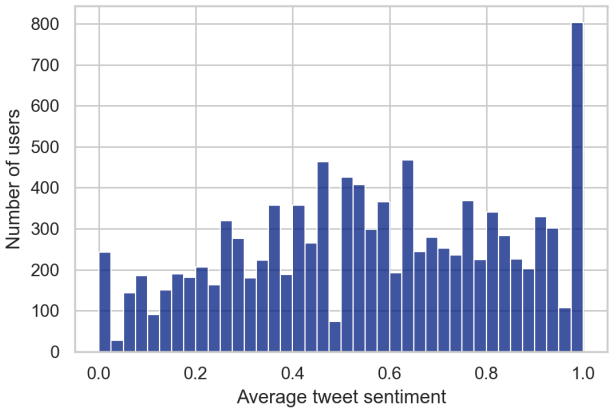

Fig. 2. Histogram representing the number of users per average sentiment

The *user* field contains the names of 10,647 distinct users; each user wrote at least 15 tweets (considering the datasets), but some users were much more prolific than others (figure 1). Analysing the user average sentiment distribution (figure 2), it is possible to notice that there are many users whose tweets are almost exclusively positive (or negative). This odd behaviour led to a deeper investigation of the texts written by each user; a mean cosine similarity among the tweets posted by the same user was computed (figure 3), and it showed that a small but significant group of users used the exact same

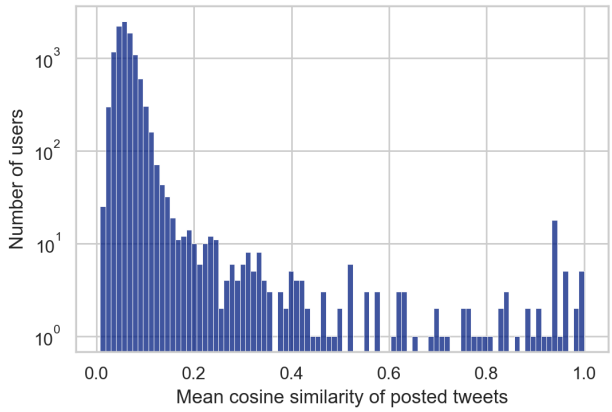

Fig. 3. Histogram representing the number of users per mean cosine similarity computed on the posted tweets of each user

words in multiple tweets. Some of the users that posted the largest number of tweets, also have the highest average cosine similarity among their tweets; a precise inspection of these users suggests that most of them behave as bots, regularly posting the same message. The problem of bot identification can be addressed using machine learning frameworks, as the one suggested by Kantepe and Ganiz [1], however for this project a more naive approach to remove suspicious users is implemented in the preprocessing step.

Finally, the *text* field collects the tweet messages. The messages are written in english, they may contain tags (i.e., starting with "@"), hashtags (i.e., starting with "#"), and URLs. HTML character entities (e.g., "*&*", "*"*") are present too.

The *text* field is the only one which is going to be considered to train the model and to predict the text sentiment; indeed, the *date* field is only useful to show that both the development set and the evaluation set refer to the same time period; instead, the *user* field is not taken into account because this would instruct the model to predict the sentiment based also on the user who posted the message, loosing the more general scope the model should have.

II. Proposed approach

A. Preprocessing

The model evaluation is made using the evaluation set as reference. As said before, the development set and the evaluation set share the same characteristics, and, presumably, the same source. Basing the model performance on this evaluation set may lead to overfitting, making the model inadequate to predict the tweet sentiment outside this project. Therefore, two possible preprocessing paths have been designed: one more complex, composed of several steps, and more suitable to a general application; and one much simpler, consisting just in the first three steps of the complex one, that performs better, but is probably less general.

The message cleaning steps composing the more articulate preprocessing path were partially influenced by [2]. The steps are the following

1) Removal of rows having the same *id* value
2) "*&*" HTML entities are removed;
3) "*"*" HTML entities are removed;
4) Words staring with @ are removed: these words are user tags, and do not give insights regarding the sentiment of the message;
5) Words staring with *http* are removed: these words are URLs, and do not give insights regarding the sentiment of the message;
6) Punctuation is removed;
7) Stemming is performed, using the Snowball stemmer by Porter [3];
8) Negations in sentences are stressed appending in subsequent words a "*_NEG*" suffix.
9) Exclusion of users behaving as bots: as seen before, some users write the same message multiple times (i.e., users that have an high average cosine similarity computed on their tweets), always associated with the same sentiment. These users should be excluded, since they "pullute" the dataset. To identify the most damaging ones, for each user a *user_suspiciousness* metric was computed[1], and the users having a value higher than 0.9 were dropped from the development set.

After these steps, the features are extracted tokeninzing the messages and creating a bag-of-words; the schema applied to weight the words is the **tf-idf** (i.e., term frequency - inverse document frequency), meaning that each word importance increases as its frequency in the same meassage increases, and lowers the more the word is used across all the messages in the collection. No maximum or minimum document frequency is set for a word to be considered. Other preprocessing options, such as the removal of stopwords, the type of normalization, and the N-Gram[2] length, are considered as hyperparameters of the preprocessing step.

B. Model selection

Two fast and light classification models were assessed:

- Naive Bayes Classifiers: it is a classifier family commonly employed in sentiment analysis contexts [4]. It uses the Bayes Theorem to predict the probability that a given feature set belongs to a particular label. The Naive Bayes Classifier family include several algorithms, in particular the Multinomial and the Bernoulli classifiers are often for document classification, with similar performance [5]. For this project, given that the tf-idf word weighting schema is used instead of a basic term occurrence, the Multinomial Naive Bayes classifier (Multinomial NB) was selected for evaluation.

[1]The *user_suspiciousness* is computed as the product between the mean cosine similarity among the user messages, and the "extremism" of his sentiment, that is the distance between the user average sentiment and a neutral sentiment (i.e., 0.5); both the mean cosine similarity and the sentiment extremism are scaled between 0 and 1 before the product.

[2]An N-Gram is a contiguous sequence of N items from a given sample of text or speech; if the items are words, N-Grams are also called *shingles*

TABLE I
CONSIDERED HYPERPARAMETERS

Preprocessing / Model	Hyperparameter	Values
TfidfVectorizer	*stopwords*	{"english",None}
	ngram_range	{(1,1),(1,2),(1,3)}
	max_features	{None,20000}
	max_df	{1.0}
	min_df	{1}
	binary	{True,False}
	norm	{l1,l2}
	use_idf	{True}
	smooth_idf	{True,False}
	sublinear_tf	{False}
Multinomial NB	*alpha*	{.01,.02,.05,.1,.2,.5,1.0}
SVC	*penalty*	{l1,l2}
	tolerance	{1e-2,1e-3}
	C	{1,10}
	max_iter	{100}

- Support Vector Machine Classifier: these algorithms carry the classification task determining the linear separators in the search space which can best separate the different classes. SVCs have already been used in sentiment analysis contexts, achieving good results [6]. Because of the extent of the dataset, in particular concerning the features, only the simpler Linear Support Vector Classifier (Linear SVC) was considered.

The best performing hyperparameter configuration was identified using a grid search.

C. Hyperparameters tuning

Both the preprocessing and the model hyperparameters were tuned. The considered hyperparameters configurations are summarized in table I. The development set is 90/10 train/validation splitted, given that the development set is fairly large, and a grid search is run on the traning subset.

III. RESULTS

The best configuration found for the preprocessing step is almost the same using the Linear SVC and the Multinomial NB, differentiating by only one parameter:

- *stopwords*: None
- *ngram_range*: (1,3)
- *max_features*: None
- *binary*: True
- *norm*: l2
- *smooth_idf*: False for the Linear SVC, True for the Multinomial NB

It is to be noted that, for both models, the grid search suggested that no words should be removed, neither using stopwords nor setting an upper limit for the features; this seems reasonable given the context of the project. Indeed, when analysing a message sentiment, dropping some words can make unclear and ambiguous the meaning of the text, reducing the algorithm classification capability.
Concerning the model tuning, the best configurations found, and the respective reached *f1 macro scores*, are the following:

- Linear SVC: best $f1\ macro \approx 0.794$
  - *penalty*: l2
  - *tolerance*: 0.001
  - *C*: 1
- Multinomial NB: best $f1\ macro \approx 0.758$
  - *alpha*: 0.1

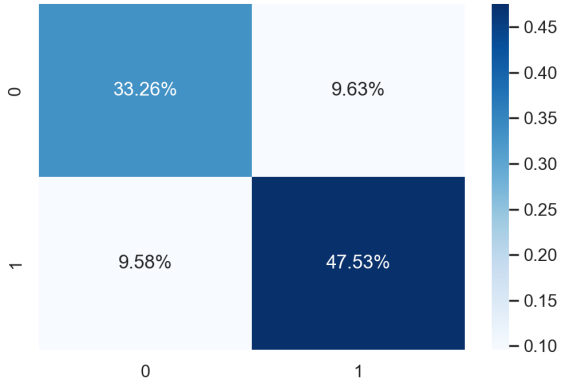

Fig. 4. Confusion matrix of the results obtained using the Linear SVC model

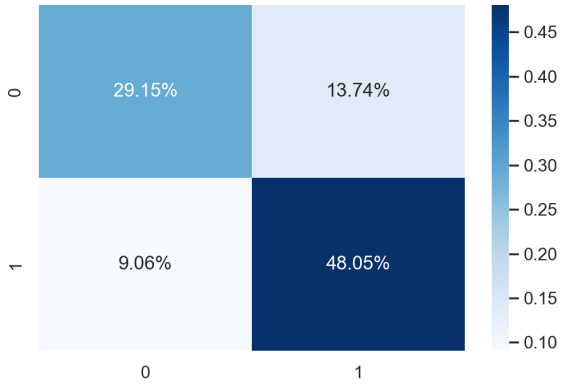

Fig. 5. Confusion matrix of the results obtained using the Multinomial NB model

After training the best performing Linear SVC and Multinomial NB on all available development data, and using these models to predict the label in the evaluation set, the obtained public scores are respectively 0.802 and 0.764.

So, it is possible to conclude that the Linear SVC model achieves better results than the Multinomial NB model. The confusion matrices in figures 4 and 5 give an insight of where the Linear SVC outperforms the Multinomial NB model: even if the latter one is slightly better in predicting the positive sentiment, it is much worse when it comes to messages with negative sentiment. This may be due to the fact that the Linear SVC model is more capable to deal with the label unbalance afflicting the development set.

To conclude, it is interesting to see how much the results are influenced by the N-Gram range parameter. This parameter

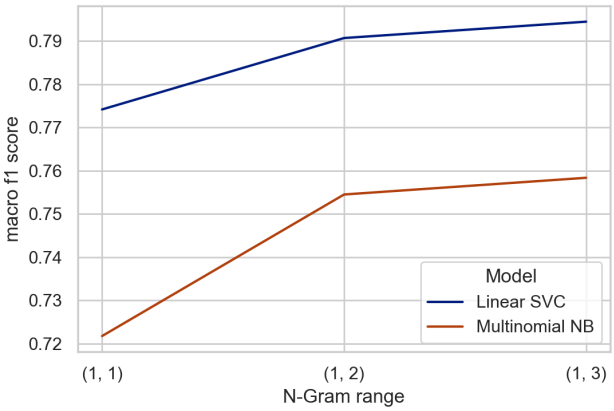

Fig. 6. Performance of f1 score for different N-gram lengths

controls the length of the N-Grams to consider during the word tokenization, so $ngram\_range = (1, 1)$ means that only single words are considered, while $ngram\_range = (1, 3)$ means that also bigrams and trigrams are included in the bag-of-words. As figure 6 shows, increasing the N-Gram length considerably improves the model results: this trend suggests the importance of word combinations to express a sentiment, and, conversely, to understand the sentiment embedded in a sentence.

IV. Discussion

Both the solutions found using the tuned models overcome the baseline score of 0.753. In particular, the Linear Support Vector Classifier reaches good results, especially considering the model simplicity and speed. However, the critical step in this project was the preprocessing: the usage of bigrams and trigrams combined with unigrams led to a wide increase in the f1-score.

Many hyperparameter configurations were left unexplored, and maybe, using a more granular grid search, it is still possible to improve the results. However, to really further enhance the obtained score, other more complex machine learning models should be considered, such as non linear SVCs or ANNs, maybe preceded by the application of a dimensionality reduction algorithm, like SVD, to reduce the number of features and to extract the most meaningful ones.

Another possible solution to explore is the usage of deep learning models with a word embedding layer, that could also be based on a pretrained word embedding stack, substantially reducing the time required for the model training. The use of deep learning models in the context of sentiment analysis has been widely tackled in recent years, with good results (see Zhang et al. [7]).

It might also be interesting to apply an unsupervised machine learning method to the dataset and compare its results with the supervised methods. To this end, a promising candidate model is VADER, that was designed specifically for sentiment analysis, and is sensitive to both polarity (positive/negative) and intensity (strength) of emotions expressed in texts; the usage of this model in a context analogous to the one of this project is discussed by Elbagir and Yang [8].

References

[1] M. Kantepe and M. C. Ganiz, "Preprocessing framework for twitter bot detection," in *2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 630–634, 2017.

[2] N. Zainuddin, A. Selamat, and R. Ibrahim, "Twitter feature selection and classification using support vector machine for aspect-based sentiment analysis," in *Trends in Applied Knowledge-Based Systems and Data Science*, vol. 9799 of *Lecture Notes in Computer Science*, (Cham), pp. 269–279, Springer International Publishing, 2016.

[3] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 40, pp. 211–218, Jan 2006.

[4] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[5] G. Singh, B. Kumar, L. Gaur, and A. Tyagi, "Comparison between multinomial and bernoulli naïve bayes for text classification," in *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pp. 593–596, 2019.

[6] N. Zainuddin and A. Selamat, "Sentiment analysis using support vector machine," in *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pp. 333–337, 2014.

[7] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[8] S. Elbagir and J. Yang, "Twitter sentiment analysis using natural language toolkit and vader sentiment," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 122, p. 16, 2019.