Twitter data classification

Matteotti M., Porcelli E.
Politecnico di Torino
Student ids: s294552, s296649
s294552@studenti.polito.it, s296649@studenti.polito.it

***Abstract*—In this report, we analyse in detail three possible classifiers to establish which is the best when performing sentiment analysis on Twitter, based on the F_1 score. After applying a thorough pre-process aimed at simplifying the task and speeding up the process by reducing its dimensionality, the Random Forest Classifier turns out to systematically outperform the other defined classifiers and a naive baseline method.**

I. Problem overview

Posts on social media platforms have increased in number to the point of becoming a reliable source of information when trying to understand the public opinion on a topic [1] [2] [3]. In this project, we look at data from Twitter, the most popular microblog platform. Twitter users utilize the platform to comment on issues or simply express their daily experiences and thoughts. In a simplified view, each tweet can be associated with a sentiment which is either negative or positive. Our aim is to build a classifier that is able to distinguish between these two categories based solely on the tweet's content, its creator and the publishing date.

The dataset is a collection of tweets published on Twitter between April and June of 2009 by English speakers. It does not contain missing values and each row of the dataset is described by the following attributes:

- *sentiment*: classification label, defined as the tweet's sentiment and coded as a binary variable. **0** when negative, **1** when positive
- *id*: tweet's unique identifier
- *date*: tweet's publishing date
- *username*: user's public name, in the format "@username"
- *flag*: query used to collect the tweet
- *text*: tweet's content

The dataset is divided into two spreadsheets:

- development set: consists of 224994 record, each labeled with the corresponding *sentiment*.
- evaluation set: consists of 74999 unlabeled record.

The classification models are all trained on the development set to correctly label the evalatuation's tweets.

II. Proposed approach

In order to find the best classifier for this task, we tried to fit a logistic regression [4], a random forest [5] and support vector machine [6] classifier, on the preprocessed dataset, as shown later in this section.

A. Data Preparation

The *flag* column can be dropped without worries as it contains the value "NO_QUERY" for every single instance of the dataset. The same holds for *ids*.

The *user* column can contain valuable information, as the user's sentiment history can ease the classification task. To understand why, one need only think of the enormous spread of *trolls* on social medias, *i.e.*, users behaving against any common social rules, with the only aim of provoking and bullying. If present, *trolls* will be more likely to post negative content compared to the average user [7]. At the same time, motivational speakers' profiles will be characterised by a more positive attitude. This possible imbalance in the distribution is shown in Figure 1 for the first 20 users in the development set.

The username itself, however, is not incredibly informative. Therefore we have transformed the *user* column into an integer, keeping track of the difference between the number of positive tweets and the number of negative tweets. In this sense, for example, *trolls* will obtain a highly negative score.

Concerning the column *date*, we performed a visual evaluation of its correlation with the sentiment variable, since we could not rely on χ^2-test due to its extreme sensitivity to sample size. Figure 2 shows that the number of positive and negative tweets per day is independent from the day itself until June 17th, 2009. From then onward, all tweets' sentiments become negative and the classification task becomes trivial. This particular pattern can be useful when training the model, and so *date* was kept and transformed from the DD-MM-YYYY format to an integer representing the tweet's distance (in number of days) from day zero, set to be the oldest tweet in the whole dataset, in this case April 6th, 2009.

B. Preprocessing

This is the essential step, as with every other Natural Language Process (NLP) task. The following preprocessing steps have been thoroughly applied to every tweet in the dataset:

1) **Case normalization**: transformation of every character into lower case;
2) **HTML entities handling**: possible HTML entities (*e.g.* the ampersand symbol "&" appears as "&") are mapped back to their actual character. Though not crucial, this step improves the human readability, enabling us to more easily perform some of the upcoming steps, especially the emoji handling;

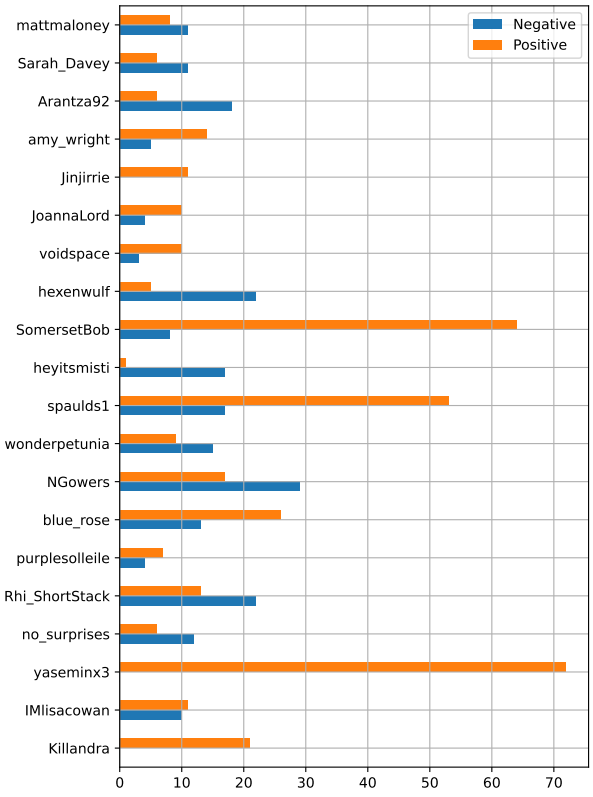

Fig. 1: Distribution of negative and positive tweets for the first 20 users in the development set. The two quantities are not always balanced.

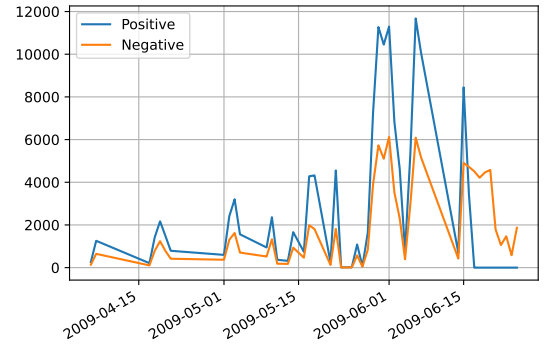

Fig. 2: Number of positive and negative tweets per day

3) **Links and username removal**: URLs and mentions are removed as they provide no information to the tweet's sentiment;
4) **# removal**: the hash is removed, but the hashtag's content is not, since it contains valuable information (*e.g.* #kobe → kobe):
5) **Emoticons handling**: substitution of emoticons with their simplified meaning: emoticons representing positive emotions have been substituted with the word "HAPPY" and the ones representing negative emotions with the word "SAD", as shown in Table I;
6) **Punctuation removal**: removal of any residual form of punctuation, since what is left from the previous step are indeed pure punctuation marks;
7) **Repeated letters removal**: removal of any repeated characters from a word. In social media, users tend to emphasize some words by repeating some of their letters, *e.g.* 'noooooo', generating potentially infinitely many ways to write the same word. We have decided to replace any character that is repeated at least 3 times with just one occurrence. Letters that are repeated only 2 times are left unchanged, because in principle it is not possible to distinguish whether they are just a repetition (*e.g.* "noo" stands for "no") or not (*e.g.* "bee" and "be" have two completely different meanings);
8) **Rare words removal**: removal from the word-matrix of all words that appear in the whole dataset at most 3 times, which are in general ortographic errors, misspellings or meaningless words. Note that this is the only step in which words are actually removed from the dataset, since stopwords are indeed kept. We made this choice because up until November the 8th 2017, all tweets were limited to 140 characters, meaning that every word is a valuable piece of information not to be wasted;
9) **Tokenization**: split of each tweet into tokens, *i.e.*, individual words, and creation of the word-matrix;
10) **Vectorization**: transformation of the word-matrix into vectors. It has been implemented using the *TF-IDF* [8] [9] matrix representation of the tweets;

In addition to that, we also ran another version of our proposed approach, which included **stemming** - the reduction of every word to its stem using *PorterStemmer()*, present in Python's *Natural Language Toolkit* [10]. This was to check whether stemming improved the performance of the classification or not.

Emoticons	Text Substitute
:-) :) ;) :o) :] :3 :c) :>:-b :b <3 =] 8) =) :} :^) :-D :D 8-D =3 =-3 8D x-D xD X-D XD =-D =D :-)) :'-) :') :* :^* >:P >:) >;) >:-) :-P :P X-P x-p xp XP :-p :p' =p	HAPPY
:L :-/ >:/ :S >:[:@ :-(:[:-|| =L :< :< :-[:-< =\\ =/ >:(:(:'-(:'(:\\ :-c :c :{ >:\\ ;(>.< >.> <.< >_< >_> <_<	SAD

TABLE I: The transformation applied to the emojis in the dataset. The emojis are taken from an article by Rajesh Kuman and Sunganya, and some modifications were made according to the database content. [11]

C. Model selection

Different algorithms have been tested and all considered as possible go-to solutions in the context of NLP tasks:

1) *Logistic regression (LR)*: endowed with strong statistical foundations, it is highly interpretable and not particularly computationally demanding;

2) *Support vector machine (SVM) classifier*: though hard to interpret and computationally expensive, the drawbacks are outweighed by generally outstanding performances.
3) *Random Forest*: though not as easy to interpret as the "simple" Decision Tree classifier, it is in general less prone to over fitting and so expected to perform better;

LR was trained using the default values. For Random Forest and SVM, the best working configuration of hyperparameters has been retrieved by applying a gridsearch explained in next section.

D. Hyperparameters tuning

We mainly focused on tuning the Random Forest and the SVM hyperparameters, since they show more room for a significative improvement. The tuning has been performed by running a gridsearch on a 80/20 train/test split of the development set (as summarised in Table II) and each performance is evaluated on the basis of the resulting F_1 score. Due to a limited computational power of our machine, an exhaustive gridsearch on SVM was not possible, since certain combinations of the hyperparameters lengthen execution times up to 12 hours. This and the fact that the few considered configurations systematically performed significally worse than all the other classifiers, made us abandon SVM without any further attempt.

Method	Parameter	Value
SVM	*C*	{0.1, 1, **10**}
	kernel	{**rbf**, poly}
Random Forest	*criterion*	{entropy, **gini**}
	n_estimators	100
	max_features	{log2, **sqrt**}

TABLE II: The hyperparameters' gridsearch. The best tested working configurations are given by the values in bold.

In a Random Forest, however, *n_estimators* could be arbitrarily increased to obtain an increasingly better performance, at the cost of a longer execution time. To find the best trade-off we ran a second gridsearch only on the Random Forest, trying fifteen different values of *n_estimators* between 1 and 500 whilst keeping all other parameters as specified by the first gridsearch. The set of tested candidates is specified in Table III.

Method	Parameter	Value
Random Forest	*n_estimators*	{1, 2, 3, 5, 10, 15, 20, 30, 40, 50, 60, 75, 100, 200, 500}

TABLE III: The candidate values for the *n_estimators* parameter in a Random Forest with {*criterion*: gini, *max_features*: sqrt}

III. Results

LR ran with default parameters achieves F_1 score ≈ 0.830, whereas SVM in its best working configuration ({*C*: 10, *kernel*: rbf}) performs surprisingly badly, achieving the worst F_1 score (≈ 0.791) amongst all the classifiers.

The proposed working configuration for the Random Forest is {*criterion*: gini, *n_estimators*: 200, *max_features*: sqrt}. Though in principle not as good as an equivalent Random Forest with *n_estimators*=500, this choice is justified by the *n_estimators*' gridsearch, whose results are summarised in Figure 3 and Figure 4. The F_1 score curve starts plateauing off at a value of *n_estimators* around 40 and the positive difference between consecutive values becomes less and less significative after *n_estimators* = 200. This configuration outperforms the other approaches, achieving an F_1 score ≈ 0.844.

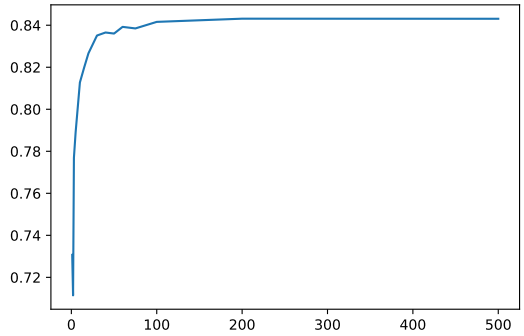

Fig. 3: Achieved F_1 score for different values of *n_estimators* in a Random Forest with {*criterion*: gini, *max_features*: sqrt}, plotted as a line plot.

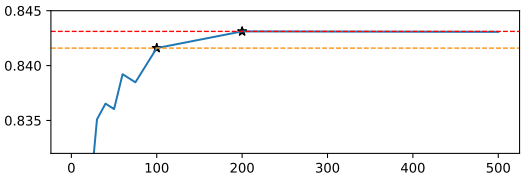

Fig. 4: Particular of the F_1 score for values [100, 200] of *n_estimators*.

Equivalent versions of LR and Random Forest classifiers including stemming as part of the preprocessing were run, and in each case the F_1 score has a small but significant decrease, as reported in Table IV. Stemming was therefore not included.

Classifier	Approximate difference between the F_1 scores with and without stemming
Logistic Regression	-0.003
Random Forest	-0.003

TABLE IV: The difference in the F_1 score with and without stemming, approximated to 3 decimal digits. A negative difference implies a decrease in the overall performances. Though small, all deltas are negative.

In light of the above, the best performing Random Forest has then been trained on the entire development set, achieving a public score of 0.836. This result is reasonably close to the

F_1 score obtained during the hyperparameters tuning, proving that our proposed approach is robust to overfitting.

For comparison, we have also fitted an equivalent baseline Random Forest trained only onto the tweets' text after having been transformed through steps 9 and 10 of the preprocessing. This method obtained a public score of 0.768. The improvement of our approach is remarkable, and it should be noted that it has also significantly reduced the number of features, speeding the execution up by hours (the baseline's training takes ∼6h30m, while the proposed approach's ∼90m).

IV. Discussion

The performance of the Random forest classifier is summarised by its confusion matrix in Figure 5. The proposed model has an unbalanced distribution of prediction errors, as it is more prone to commit type I errors. The low frequency of false negatives suggests that the disappearance of positive tweets after Jun 17th is likely to be present in the evaluation set as well, justifying our choice of keeping *date* in the model training phase.

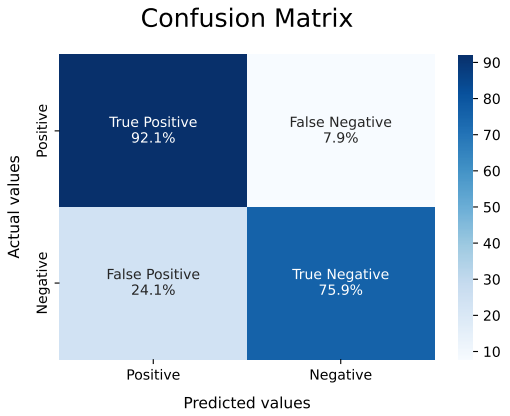

Fig. 5: Confusion matrix for the random forest with optimal parameters configuration

The poor performance of SVM can be interpreted by looking at the dimensionality of our problem. The model is trained on a dataset of dimensions 299993×23119. Even though SVM is amongst the principal methods to deal with any classification tasks, some limitations arise when dealing with large data sets [12] since the hyperplane that generates the best separation is retrieved by solving the Quadratic Programming.

Regarding the negative effect of stemming, our results in fact agree with that shown by Bao et al [13], that is a small but significant decrease in the overall performance when used in the context of Twitter Sentiment Analysis.

Although the final result is decent, we would have expected the preprocessing to have a larger impact on the overall score. We suspect that the greatest limitation of our approach is Twitter's intrinsic structure. Being an informal public space with limited characters, users will pay more attention to the content than the form integrity, making use of, for example, abbreviations and slangs. In this sense, each of the most common NLP techniques cannot be as precise as they would be on a formal essay. In this sense, the information about *user* and *date* turned out to be extremely valuable, significantly improving the overall performance.

Future work on this task should include a more exhaustive hyperparameter tuning phase that was not possible due to time limitation. In addition to that, a different SVM classification approach could be investigated. A possible solution is proposed by Cervantes et al [14], whose technique based on minimum enclosing ball clustering has been demonstrated to perform better than classic SVM when dealing with large datasets. Moreover, advanced models based on Deep Learning could also be considered. Though extremely expensive computation-wise, with the right architecture they are expected to outperform the random forest classifier.

References

[1] B. Joyce and J. Deng, "Sentiment analysis of tweets for the 2016 us presidential election," in *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pp. 1–4, 2017.

[2] S. Mansour, "Social media analysis of user's responses to terrorism using sentiment analysis and text mining," *Procedia Computer Science*, vol. 140, pp. 95–103, 2018. Cyber Physical Systems and Deep Learning Chicago, Illinois November 5-7, 2018.

[3] L. Martin-Domingo, J. C. Martín, and G. Mandsberg, "Social media as a resource for sentiment analysis of airport service quality (asq)," *Journal of Air Transport Management*, vol. 78, pp. 106–115, 2019.

[4] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[5] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1, pp. 278–282, IEEE, 1995.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] H. F. Gylfason, A. H. Sveinsdottir, V. Vésteinsdóttir, and R. Sigurvinsdottir, "Haters gonna hate, trolls gonna troll: The personality profile of a facebook troll," May 2021.

[8] L. H. Peter, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of Research and Development*, vol. 1, pp. 309–317, 1957.

[9] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

[10] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[11] B. Rajesh Kumar and S. Sunganya, "Collecting emoticons and stored in data base using unicode via python," *Internation journal of scientific technology research*, vol. 9, no. 3, pp. 4161–4164, 2020.

[12] J. xiong Dong, A. Krzyzak, and C. Suen, "Fast svm training algorithm with decomposition on very large data sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 603–618, 2005.

[13] Y. Bao, C. Quan, L. Wang, and F. Ren, "The role of pre-processing in twitter sentiment analysis," in *Intelligent Computing Methodologies* (D.-S. Huang, K.-H. Jo, and L. Wang, eds.), (Cham), pp. 615–624, Springer International Publishing, 2014.

[14] J. Cervantes, X. Li, W. Yu, and J. Bejarano, "Multi-class support vector machines for large data sets via minimum enclosing ball clustering," pp. 146 – 149, 10 2007.