Twitter Sentiment Analysis - CS 583 - Data Mining and Text Mining

Emilio Ingenito - University Of Illinois Chicago - UIN: 656518117

Abstract - In this report it is proposed an approach for sentiment analysis classification based on Twitter posts. The chosen approach includes data analysis, pre-processing and feature extraction to better characterize the data. Multiple classification models were tested, including neural network, in order to get the best possible results.

1 PROBLEM OVERVIEW

1.1 Introduction

The project aims to analyze a dataset that contains a collection of tweets in tabular format. The purpose is to predict the sentiment of a given post on Twitter. The sentiment provided in this task is either positive (1), negative (-1) or neutral (0). In the provided dataset there is also a category *2* of mixed sentiment, both positive and negative, that won't be considered in the analysis.

The dataset is provided as two CSV files containing Twitter posts about Obama and Romney in different sheets:

- Obama dataset containing 7198 observations
- Romney dataset containing 7200 observations

The dataset is based on three different features:

- **Text**: the text of the tweet;
- **Date**: the day, month and the year of tweet's publication;
- **Time**: the time of the day at which the tweet was published.

Note that, as a first, initial step, I removed any tweet which has null values in the *text* column, as well as those instances associated with an inconsistent label (either *2* or *!!!*).

1.2 Data distribution

Considering the whole dataset (Obama + Romney), looking at the distribution of sentiment class (Fig. 1) we have 25% of tweets related to positive sentiment, 43% for the negative sentiment, 32% relative to neutral sentiment. It can be stated that the dataset has almost balanced classes.

The situation does not essentially differ if we consider only Obama's tweets: the dataset looks, again, almost balanced (Fig. 2). On the other hand, by analyzing Romney's tweets, we can observe a slightly skewed distribution (Fig. 3, skewed towards the negative class (-1).

Is is also useful to analyze the so called *word cloud*, a useful tool which enable us to visually inspect the most frequent word of tweets associated with each of the classes. Fig. 4 shows that positive tweets are associated with words such as 'well', 'good', 'winner', 'won', 'great', 'proud' and so on, while the negative ones (Fig. 5) are associated with words such as 'lose', 'stop', 'worse', 'joke', 'issue' and so on.

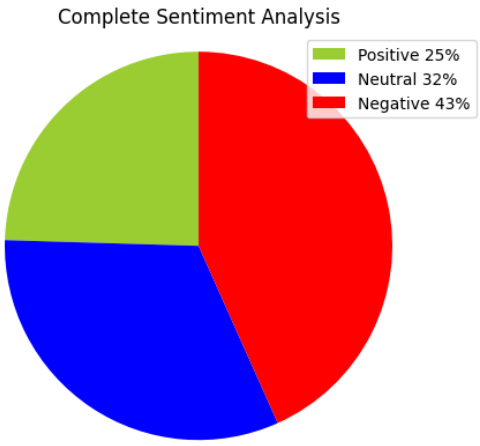

Figure 1: Class labels Obama + Romney

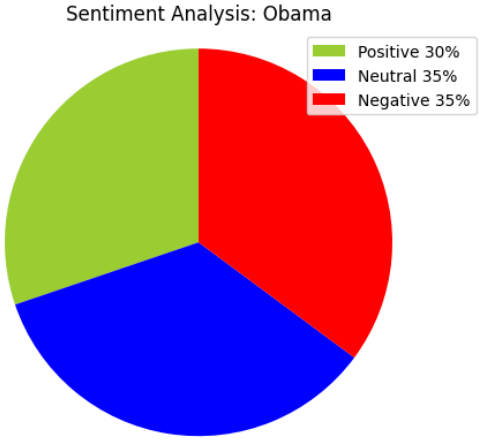

Figure 2: Class labels distribution for Obama Dataset

2 TECHNIQUES - PROPOSED APPROACH

2.1 Instance dropping

The first step taken into account was the dropping of instances which would not be useful for the future classification. The raw data provided needed preprocessing to be correctly evaluated: since those data were directly extracted from Tweeter, some of them included inconsistent information. Specifically, I decided to remove from the dataset those instances which belonged to at least one of the following categories:

- the examples containing dates that were not valid;
- the examples containing a sentiment with a value not in {-1,0,1};
- the examples with missing text data.

2.2 Text Cleaning

Then, the second step taken into account was the data preprocessing. Raw tweets often include typos, acronyms, hash-

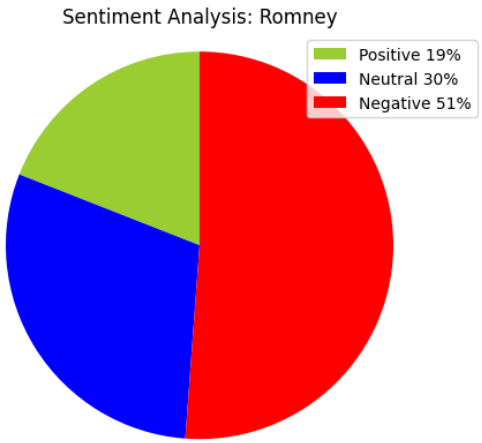

Figure 3: Class labels distribution for Romney Dataset

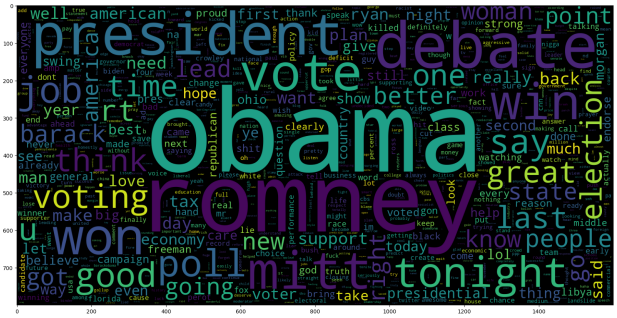

Figure 4: Word cloud for positive tweets

tags, urls, usernames preceeded by @ and many other elements which do not carry any useful information to classify those tweets. Instead, such elements could add noise and worsen the performance of the classifier. Hence, these are the steps followed to clean the dataset:

- remove urls (words starting either with *http* or *www*)
- remove emoticos (e.g. :) and so on)
- remove mentions (words starting with @)
- remove hashtags
- remove numbers and special characters
- remove tags, such as <a> and
- fix contractions and typos
- remove stopwords

Furthermore, other famous techniques were tested, in order to achieve a better classification accuracy, such as stemming and lemmatization. The latter consists of grouping together the different inflected forms of a word so they can be analyzed as a single item. Is is similar to stemming but it brings context to the words.

2.3 Weighted document representation

To extract relevant information from the textual description, I tried different vectorization techniques: each of them takes as an input the twitter's *cleaned* text and outputs an array of floating point values. This is a crucial step, as those vectors will be used as the training data for the different classifiers. The following vectorizers were tested:

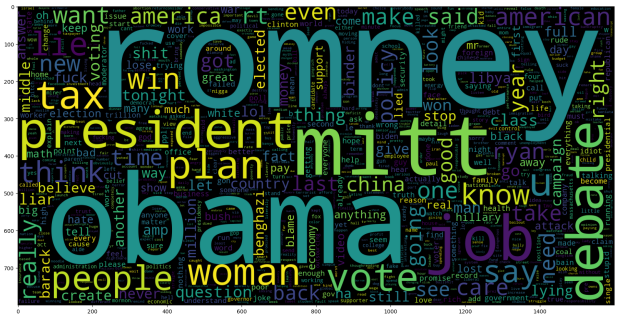

Figure 5: Word cloud for negative tweets

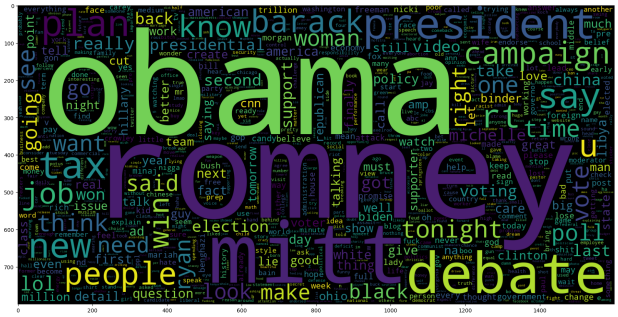

Figure 6: Word cloud for neutral tweets

- **tf-idf**: the value of each vector component is the number of times the corresponding term occurs in the document, multiplied for a specific weight. The weighting technique is the one relative to the term frequency-inverse document frequency (tf-idf), suitable for heterogeneous documents.
- **bag of words**: this techniques just consists of counting the frequency of words in a document. Thus the vector for a document has the frequency of each word in the corpus for that document.
- **Word2Vec**: it is an algorithm that uses a 2-layer neural network to ingest a corpus and produce sets of vectors.
- **BERT**: it is a technique developed by Google that uses a transformer based ML model to convert phrases and words into vectors.

2.4 Feature Selection

Besides the text, I tried to exploit also the date and time information as an additional feature. Hence, the date was transformed into categorical data (the day belonged to the set {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}, Fig 8), as well as the hour (which was transformed such that it was associated with only one value of the set {morning, night, noon}, Fig. 9). Finally, such categorical data was represented through a one-hot-encoding representation.

3 TECHNIQUES - MODEL SELECTION

Sentiment analysis is an active area of research, and up to now, many approaches have been proposed. The state of the art for text classification shows that neural networks outperform every other model. In order to empirically test it, I tried some of the most famous models, along with a neural network. More specifically, the following models were tested:

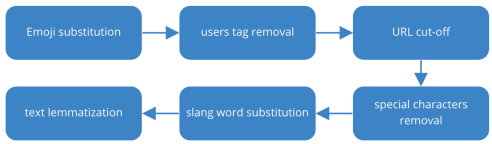

Figure 7: Scheme of the text preprocessing used

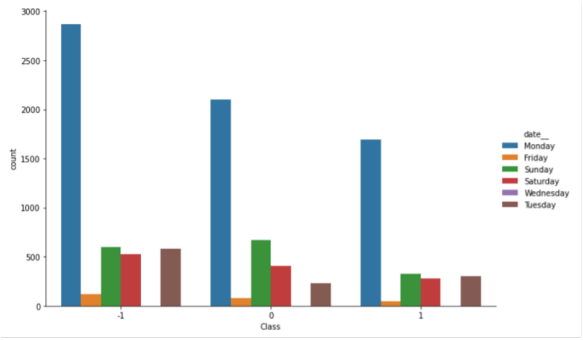

Figure 8: Tweets' distribution - based on the day of week

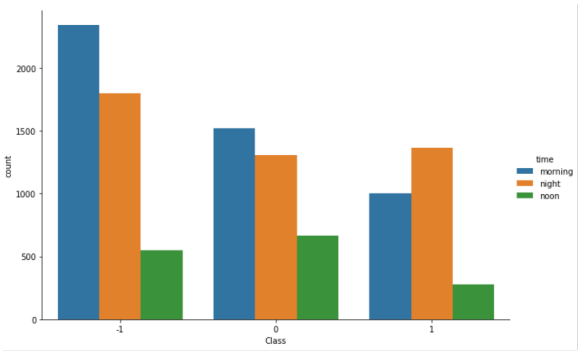

Figure 9: Tweets' distribution - based on daytime

- Support Vector Machine
- Naive Bayes
- Decision Tree
- Extreme gradient boosting
- A neural network

3.1 Support Vector Machine

SVM is a supervised-learning model frequently used in classification tasks. The aim of SVM is to find a linear hyperplane that will separate the data maximizing the margin of data split. When the datset is not linearly separable, it is crucial to exploit the right kernel function, in order to get the best results. In this analysis, it was tested a linear, a polynomial and an RBF kernel. The RBF (Radial Basis Function) kernel, is a kernel that is frequently used since it typically provides precise results. All parameters were initially set default using scikit-learn library, later through cross-valuation were detected the optimal hyperparameters C and gamma.

3.2 Naive Bayes

Naive Bayes classifiers are linear classifiers that are known for being simple yet very efficient. The probabilistic model of naive Bayes classifiers is based on Bayes' theorem, and the adjective naive comes from the assumption that the features in a dataset are mutually independent.

In practice, the independence assumption is violated, but naive Bayes classifiers still tend to perform very well under this unrealistic assumption.

3.3 Decision Tree

Decision tree learning is one of the most widely used techniques for classification. Its accuracy is competitive with other methods and it is very efficient. It works by dividing a dataset into smaller and smaller subsets, based on features, until a final prediction or decision can be made. The end result is a tree-like structure where each internal node represents a test on an attribute, each branch represents the outcome of that test, and each leaf node represents a class label.

3.4 Extreme gradient boosting

Extreme Gradient Boosting (XGBoost) is an advanced implementation of gradient boosting, which is a popular machine learning technique for regression and classification problems. In XGBoost, decision trees are constructed in a sequential manner, with each tree built to correct the errors of the previous tree. It uses a gradient descent algorithm to minimize the loss function, which measures the difference between the predicted and actual values. XGBoost also includes regularization techniques to prevent overfitting, such as L1 and L2 regularization, and provides built-in handling of missing data.

3.5 Neural Network

In order to carry out the classification task through a neural network, I chose BERT (Bidirectional Encoder Representations from Transformers), a popular pre-trained deep learning model for natural language processing tasks, including sentiment analysis. Unlike traditional models that process text sequentially, BERT uses a transformer-based architecture that can take into account the context of each word in a sentence by considering the entire sequence of words at once. BERT has been shown to outperform traditional machine learning models and other deep learning models on various benchmark datasets for sentiment analysis, including the Stanford Sentiment Treebank and the IMDb movie review dataset. Here are the main characteristics of its configuration:

Model name	bert-base-uncased
Hidden activation function	gelu
Hidden size	768
Number of hidden layers	12
Vocabulary size	30522

3.6 Hyperparameter tuning

For all the classifiers, the best configuration of hyperparameters has been determined through a grid search 10-fold cross validation. In the following table are shown the considered hyperparameters.

Classifier	Hyperparameter tuned
SVM rbf kernel	C, gamma
Naïve Bayes	Fit prior
Decision Tree	Depth, criterion
Neural network	learning rate

4 RESULTS

For evaluating the performance of the models two metrics were mainly taken into account, F1 score and accuracy. Since the dataset is almost balanced, the accuracy score is a significant metric, since it provides useful information about how well the classification task has been perofrmed.
To get the final results, the 10-fold stratified cross validation was employed. This technique allows to maintain the same proportion of the classes in the folds created.
Furthermore, the given results are the best combination of the classification algorithm and the input vectorizer: this means that, exploiting the scikit-learn library, I tried all possible combinations of input vectorization - classification model, to verify which was the best vectorizer for the given model. It turned out that the Tf-Idf vectorizer was the best one with the traditional algorithms (SVM, Naive Bayes and so on), while the BERT vectorizer was the best for the neural network. As a further observation, the results obtained with and without taking into account the date/time features were quite close, so such features were discarded.
It is important to notice that just one model was produced, i.e. the Obama dataset and the Romney one were merged and considered as a single dataset.
In the next page (Fig. 12, 13) are summarized the results achieved by the different models, in terms of accuracy and F1 score.

4.1 BERT results

The best model, with regard to accuracy, appears to be the BERT based neural network: this is important, since I had an empirical experience of what has been stated multiple times, namely that neural networks usually outperform the traditional machine learning algorithms. Fig. 10 and 11 show the confusion matrix with reference to the validation and the test set, after the training phase of the neural network. As we can observe, the model has the best performances on the negative class (here associated with 2 which is an encoding for the neural network labels).

5 Conclusions

In this project several techniques were tested to perform and carry out the sentiment analysis task, specifically for assigning a sentiment (positive, neutral or negative) to some given tweets.
It was first needed to perform pre-processing, since the raw data contained useless and noisy information. This step was critical to get higher accuracy at classification time. Then, it was needed to encode the *cleaned* text through a vectorizer, and many vectorization algorithms were tested, with different approaches and different logic.
Finally, the classification task: different algorithm were tested, and also a neural network (the first time in my career). Through 10-fold cross validation I carried out hyperparameter tuning, in order to find the best possible combination.
As a last step, I evaluated the obtained results and choose the neural network as a final model to perform such classification task, based on the accuracy and the F1-score achieved on the test set.

5.1 Lessons learned

This project allowed me to dive more deeply in the data-science world. By doing some research about those topics, I realized the following, as lessons learned:

- how important it is to deeply analyzed the dataset and the provided information;
- how important it is to deeply understand which feature to be chosen to achieve the best possible results;
- how important it is to study and use the most useful machine learning libraries, which can speed up the process;
- how important it is to perform cross validation for hyperparameter tuning;
- how important it is to understand the metrics in order to evaluate the models.

5.2 Future work

Suggestions for future work would include exploring in depth other neural networks, which are the current state of the art for text classification.

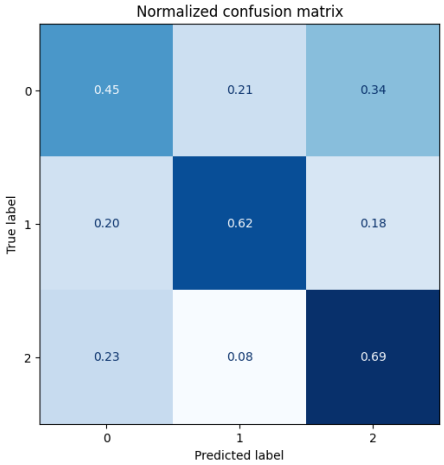

Figure 10: The confusion matrix of the BERT model on the validation set, at the end of the neural network training

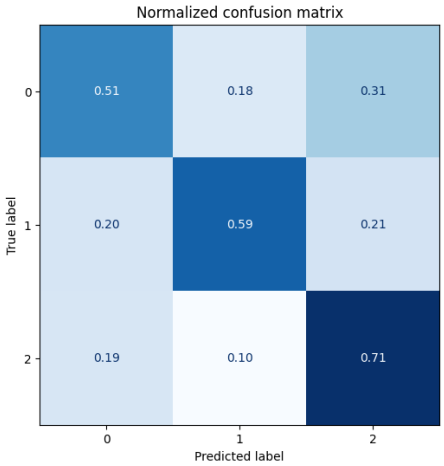

Figure 11: The confusion matrix of the BERT model on the testing set, at the end of the neural network training

Accuracy and F1 scores of the different models:

- **XGB**
- **Decision tree**
- **SVM**
- **Naive Bayes**
- **BERT model**

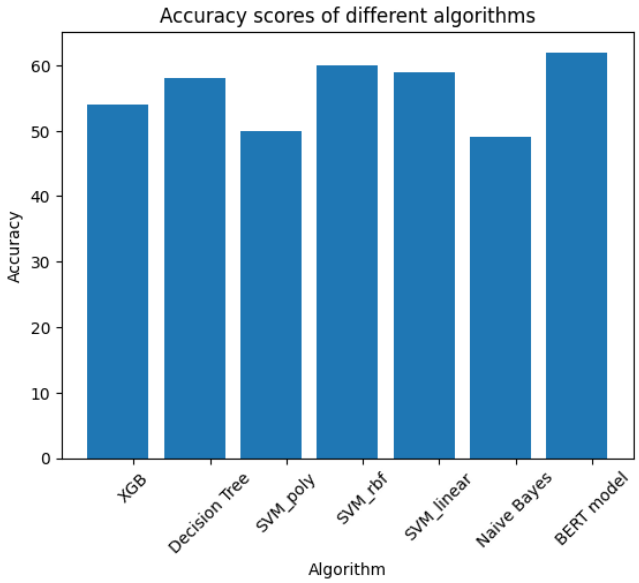

Figure 12: The accuracy score of the different models: the best one is the BERT one with an accuracy of 62%

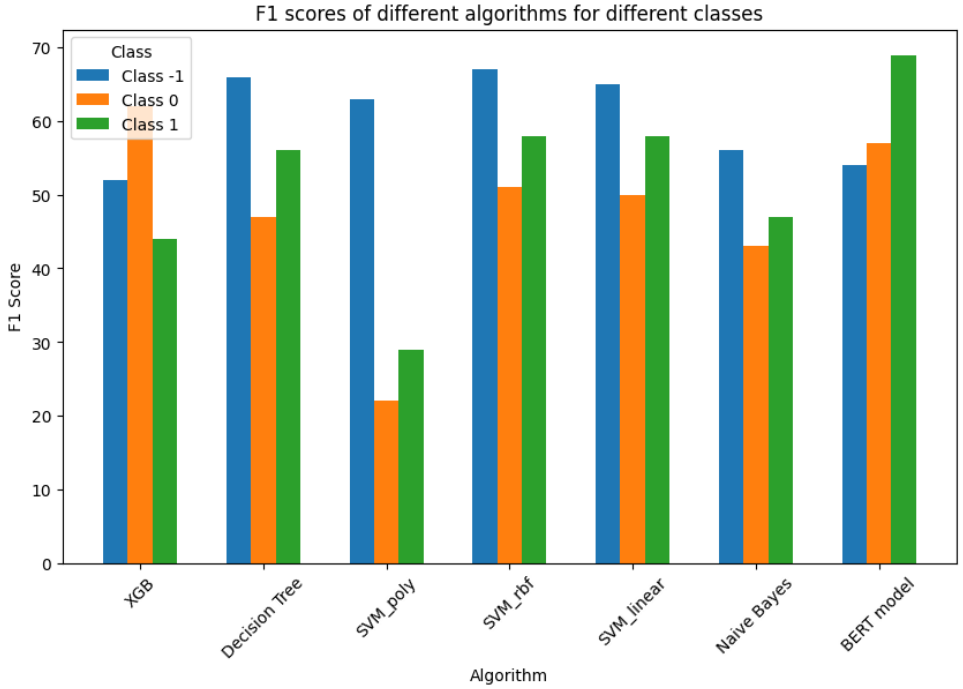

Figure 13: The F1 score of the different models, distinguishing from the different classes {-1, 0, 1}