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

Marco Colombo - University Of Illinois Chicago - UIN: 662575650
Andrea Carotti - University Of Illinois Chicago - UIN: 673145962

Abstract - In this report we propose an approach for sentiment analysis classification based on Twitter posts. The chosen approach embeds data analysis, preprocessing and feature extraction to better characterize the data. The solution compares different classification models to get the best possible result.

1 PROBLEM OVERVIEW

1.1 Introduction

The project aims is 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 dataset provided is also present 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 7200 observations
- Romney dataset containing 7200 observations

Our dataset is based on three different features:

- **Text**: Which contains the text of the tweet
- **Date**: Which contains the day, month and the year of publication of the tweet
- **Time**: Which contains the time of the day at which the tweet was published

1.2 Distribution of data

Considering the whole dataset (Obama + Romney), looking at the distribution of sentiment class (Fig 1) we have 24.48% of tweets relative to positive sentiment, 40.83% for the negative sentiment, 32.53% relative to neutral sentiment and 2.16% undefined due to the rawness of the data (mixed sentiment is excluded). We conclude that the dataset has almost balanced classes.

1.2.1 Obama tweets

Considering only the tweets about Obama, as can be seen from the figure 2, the dataset is not skewed.

1.2.2 Romney tweets

Differently from Obama, Romney's tweets reveal a distribution slightly skewed (Fig 3). Most of the tweets were indeed classified as 'negative' (-1). And many tweets contained values different from the three classes -1, 0, 1. So the dataset revealed to be slightly skewed towards the negative class.

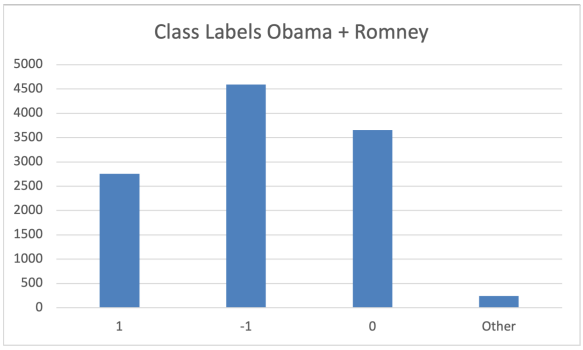


Figure 1: Class Labels Obama + Romney

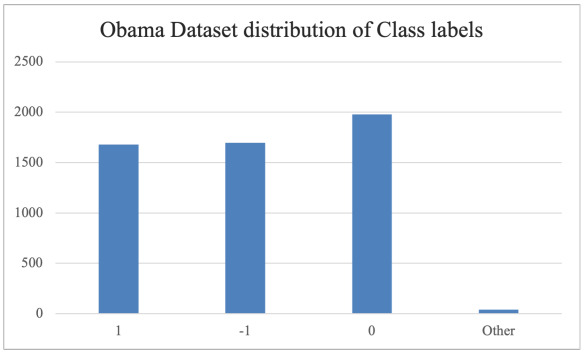


Figure 2: Class Labels distribution for Obama Dataset

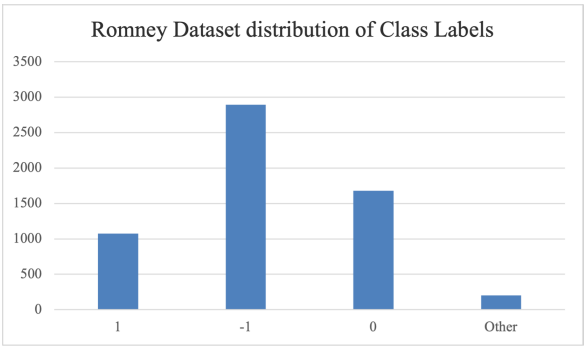


Figure 3: Class Labels distribution for Romney Dataset

2 TECHNIQUES - PROPOSED APPROACH

2.1 Text Cleaning

Since we are dealing with textual data, we need to process the documents, in order to get the set of features that will be used in our classification model. For this reason, the phase of preprocessing and data cleaning is fundamental in this project. First of all, we transform our tweets in order to have a cleaned version of our data. We substitute the emoji according to

their meanings (sad, neutral, or happy), we change all users' tags and hashtags by inserting the word 'tag' and 'hashtag' respectively; we cut off the URL, the numbers and the different punctuation. In the text are present special characters due probably to the collection of data from internet sources, like the string '"' or '&' that represent respectively '"' and 'and'. The following step consists in changing all slang words, text abbreviations and special characters with their sense. To reach this result we got a list of possible slang and chat abbreviations from a website, combining them with other abbreviations that we got looking to the records. Finally, we do the lemmatization of the text, which consists of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words.
We removed the <e> and </e> tags every time they appear in the text.

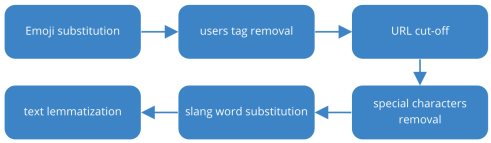


Figure 4: Scheme of the text preprocessing used

2.2 Weighted document representation

To extract relevant information from the textual description, we implement the tf-idf representation. Through the tf-idf vectorizer, we transform the documents into term vectors. Each term is a component of the vector and the value of each 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, where

$$tf\_idf(t) = freq(t,d) * log(m/freq(t,D))$$

With $freq(t,d)$ we represent the frequency of the term t in the specific document d, while the logarithm has the role of weight the frequency according to the number m of documents in which the term appears. This allows us to use the different words as separate features. Due to the big amount of tweets and different words we decided to let the td-idf matrix in a sparse format.

2.3 Samples Dropped

As we said before, the raw data provided needed preprocessing to be correctly evaluated. Since some of the data were missing or they were incomplete, we decided to drop all the samples where this happened, or where data had inconsistent value. The choice was made because the proportion of data dropped, with respect to the total number of samples was still an insignificant amount.

From the original dataset the following samples were dropped:

- All the samples containing dates that were not valid
- All the samples containing a sentiment with a value not in {-1,0,1}
- All the samples with missing data

2.4 Date Cleaning

Since the data provided was raw, also dates needed to be preprocessed. First we transformed the dates in the standard format month-day-year. Then we obtained the one-hot encoding representation of dates as days of the week. Through this process categorical data were converted into numerical data, to be correctly evaluated by the models used. Days of the week were chosen instead of month of the year because a given month could be more subject to overfitting due to specific events occurred in that same month, instead it is likely that people's sentiment depends on the day of the week.

Each tweet was grouped depending on the day of the week it belonged ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday').

2.5 Time Cleaning

The same approach (one-hot-encoding) was used in order to process the time of publication of the tweets. First, we transformed the time provided in raw format into hours-minute-seconds format. Then, we grouped according to the hour in which each tweet was posted. But since this approach didn't improve the performance we then grouped the tweets in 3 main time of the day 'morning', 'noon' and 'night'.

2.6 Feature Selection

In addition to the tf-idf representation, where each word becomes a feature, we also tried to use the date and time features. We found out that, given a sentiment, the number of examples are not completely evenly distributed across the three possible values (morning, noon, night) of the feature time of tweet (Fig 6); similarly, the same holds for the feature day of week (Fig 5). As a result we tried to include these features in our analysis to see whether the performance of the classification improved or not.

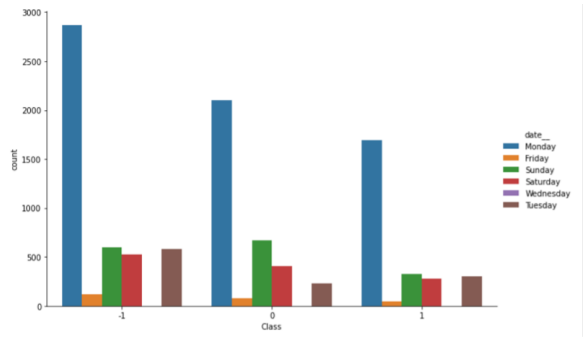


Figure 5: Distribution of the tweets based on the day of week of their publication

Sparse matrices were employed to handle both the tf-idf representation and these extra features. In fact, using a sparse representation allows to drastically reduce the computational

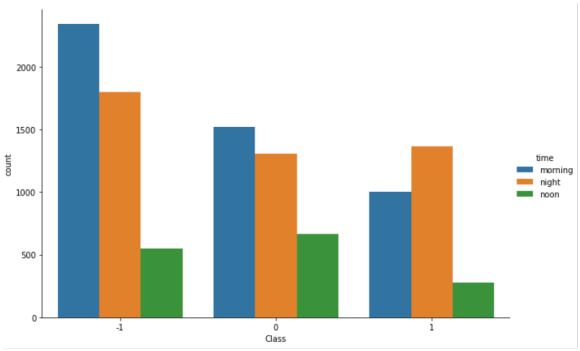


Figure 6: Distribution of the tweets based on the time of the day of their publication

complexity for training and testing the models (by using a dense representation of the matrices, the training and testing time was several order of magnitude greater).

2.7 SMOTE

Since the dataset resulted to be slightly skewed towards the negative class, we tried also to employ SMOTE data augmentation to balance the classes in order to improve performance of our classifiers.

SMOTE is an algorithm that performs data augmentation by creating synthetic data points based on the original data points. This can be seen as an advanced version of oversampling. In particular, we decided to resample all classes but the majority class. In this way, all the classes have the same number of examples of the number of examples of the majority class. We chose this technique instead of performing undersampling because it would have reduced too much the number of available examples.

3 TECHNIQUES - MODELS SELECTION

Sentiment analysis is an active area of research, and up to now, many approaches has already demonstrated to outperforms others. From the research conducted we selected the models (excluding neural networks) that proved to have the best performance:

- Support Vector Machine
- Logistic Regression
- Naive Bayes
- Decision Tree

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.

An essential thing for SVM is the choice of the kernel function. Therefore, in our analysis we tried both the linear svm model and the svm with the RBF kernel.

The RBF (Radial Basis Function) kernel, it is a kernel that is frequently used since it typically provides precise results.

RBF kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$$

All parameters were initially set default using scikit-learn library, later through cross-valuation we detected the optimal hyperparameters C and gamma.

3.2 Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

In our analysis multinomial logistic regression was used.

3.3 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.

3.4 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. The classification model is a tree, called a decision tree. The tree is constructed in a top-down recursive manner, at start, all the training examples are at the root, then examples are partitioned recursively based on selected attributes. Attributes are then selected on the basis of an impurity function.

3.5 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
Logistic regression	Penalty

4 RESULTS

For evaluating the performance of our models we have used two main metrics: F1 score and accuracy. In particular the F1 score is crucial in this context since it gives information about both precision and recall of the classifiers for each class, while accuracy does not take into account the difference among the three sentiments we are interested in.
To get the final results, we employed the 10-fold stratified cross validation. We used this technique because the classes are not perfectly balanced, and so it allows to maintain the

same proportion of the classes in the folds created. Furthermore, we used the Scikit-learn Pipeline to sequentially apply the tf-idf vectorizer and the final estimator. This is very important because without the use of this technique, we would have incurred into overfitting. Indeed, for each fold of the cross validation, it is important to learn the vocabulary and idf from training set and then transform the tweets of the test set using the vocabulary and document frequencies learned from the training set.

We kept the results obtained with both the extra features (date and time) and without them to compare the two different approaches.

In addition, different performances were obtained depending on the dataset considered:

- Obama Dataset only
- Romney Dataset only
- Obama and Romney dataset together

Finally, we evaluated also the performances obtained by using the added synthetic data points by the SMOTE function.

Regarding Obama dataset, we had the best results with the svm-rbf considering only the text (with tf-idf) as feature, and the logistic regression considering also the date and time features. In this case the dataset was already almost balanced, so the oversampling did not provide any improvement to the results. Regarding Romney dataset, we had good with the svm-rbf considering only the text (with tf-idf) as feature, even if none of the techniques outperformed all the other ones. In this case, the oversampling provided a better balance between the f1-score of positive and negative class. This can be seen for example by looking at the difference of the f1-scores of Romney, considering only the text with and without resampling using the Naive Bayes classifier. However, as said before, there is not a combination of (classifier, features used, resampling yes/no) that completely outperforms the other ones, but each case is a trade-off between the accuracy, the f1-score of the positive class and the f1-score of the negative class.

In the following pages are shown the bar charts of accuracy and f1-score for the different classifiers, in case of all features used (useOnlyText: false) or in case of the usage only of the text (true), and in case of the usage of oversampling or not.

5 Conclusions

In this project we tried several techniques to perform our sentiment analysis task, specifically for assigning a sentiment (positive, neutral or negative) to some given tweets. We first focused on the text preprocessing phase, fundamental in this application, and we used the tf-idf weighting scheme. We also tried to use the additional features available from the dataset, date and time, and we tried to use SMOTE for performing over-sampling in order to balance the dataset. To perform the classification task we used different classifiers among the most promising ones for sentiment analysis.

From this project we learned a lot since it was our first time dealing with a supervised learning task. In particular, these are the main lessons we learned:

- Importance of analyzing dataset before starting working on the next steps.
- Importance of feature selection and creation.
- Importance of looking for documentation from domain experts to have a better knowledge of the most promising techniques.
- Importance of cross validation for hyperparameter tuning, it is not easy to choose the values.
- There is not a unique solution: intuitions, experience and knowledge are fundamental to obtain good results.

Suggestions for future work include trying new embeddings to represent text and the usage of deep learning techniques. These ones nowadays are the most promising approaches to address this problem.

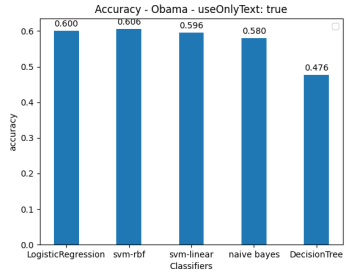


Figure 7: Accuracy in the Obama dataset without using SMOTE data augmentation and without date and time

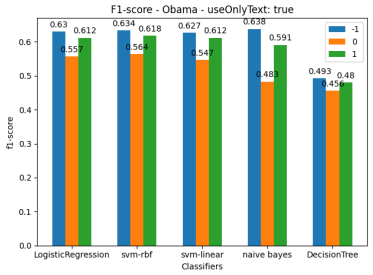


Figure 8: F1-Score in the Obama dataset without using SMOTE data augmentation and without date and time

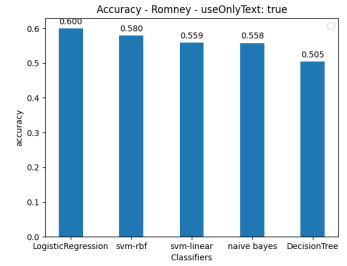


Figure 9: Accuracy in the Romney dataset without using SMOTE data augmentation and without date and time

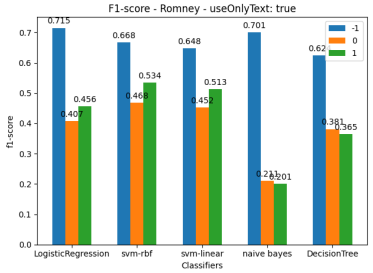
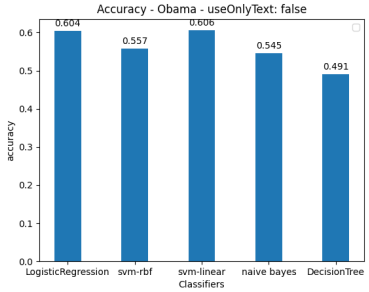
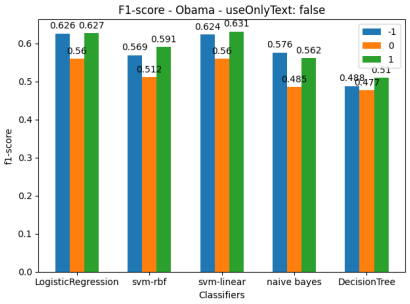
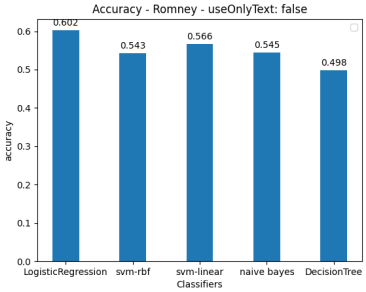


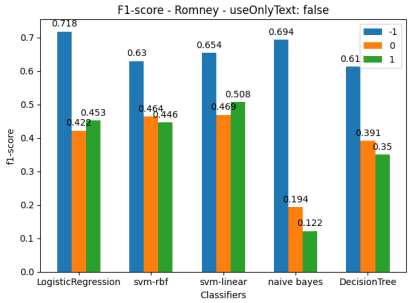
Figure 10: F1-score in the Romney dataset without using SMOTE data augmentation and without date and time

Performance of the models without resampling (no data augmentation using SMOTE) and using also date and time features:

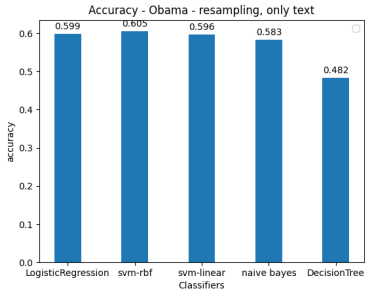


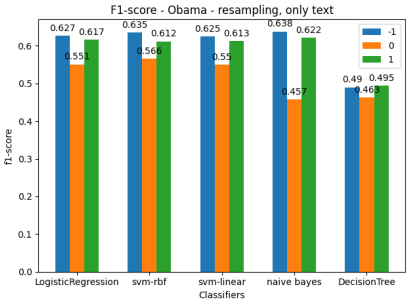


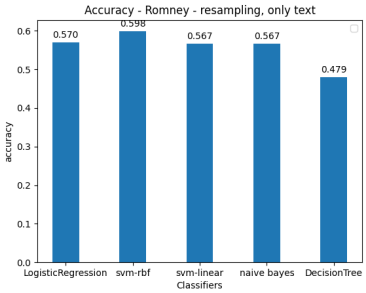


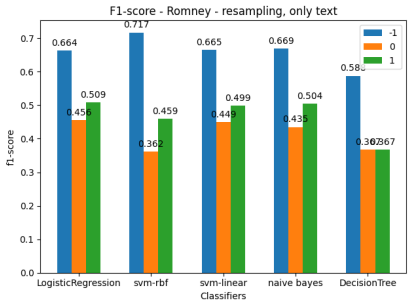


Performance of the models with resampling (including data augmentation using SMOTE) and without using date and time features:









Performance of the models with resampling (including data augmentation using SMOTE) and using date and time features: