Twitter sentiment analysis using supervised Machine Learning

Rafael Soares
ISCTE-IUL
Mestrado em Engenharia Informática
Text Mining
Docentes: Fernando Batista, Ricardo Ribeiro

Tomás Machado
ISCTE-IUL
Mestrado em Engenharia Informática
Text Mining
Docentes: Fernando Batista, Ricardo Ribeiro

***Abstract*—Due to the high amount of data that is generated on a daily basis by Social Network users with their personal opinions on different topics and reviews on products, Natural Language Processing and Text Mining come to take advantage of this data in order to reach conclusions for a high variety of purposes. This article focus on data generated by Twitter and seeks for a better understanding of the most efficient methodologies to extract the sentiment classification for the data. To do so, various experimentations were performed, with a variation of pre-processing methods, use of a sentiment lexicon model and Naive Bayes, Support Vector Machine and Logistic Regression supervised Machine Learning algorithms to classify if a given tweet is either a positive or a negative sentiment. Even thought the remotion of StopWords is considered important in the pre-processing phase and high accuracy levels are reached by Naive Bayes and Support Vector Machine algorithms is checked in the Related Works section, our results showed better accuracy levels with Logistic Regression algorithm without the remotion of StopWords.**

I. Introduction

We live in times where the use of social networks are increasingly present in the daily base of our lives. A social network is a web-based service where people can communicate, share ideas and interests with each other. One of the reasons for the importance of social networks is the fact that users tend to express their opinion on a particular matter or evaluate a product. As more users share personal opinions on different topics and personal reviews on products, social networks are increasingly becoming a valuable source of information for the opinions and sentiments of its users [1]. For this reason, natural language processing and text mining tools are starting to gain importance as they can take advantage of these source of informations to gather different opinions on different topics from the social network users and reach different conclusions.

Our work is based on Twitter data, as it is one of the main social networks where users express their opinions on a regular basis. Sentiment Classification of opinions, represents a level of aggregation that can be used for multiple purposes, such as social studies, political interests, adjusting marketing strategies and market researching. To do so we conduct Text Mining and Sentiment Classification experiences using different types of pre-processing methodologies. Further we apply, Machine Learning Algorithms in an Supervised way, such as, Support Vector Machine, Naive Bayes and Logistic Regression supervised machine learning algorithms mentioned in the following sections to understand how the results on emotion classification can vary.

II. Related Work

There are several articles from the past years looking to explain how to obtain the best results from sentiment and opinion analysis of users based on tweets. Go et al analyzed tweets for positive and negative emotions. In the article is explained the importance of emoticons in Twitter such as ':-)' and ':-(' and how they help classify the sentiment of a tweet as a positive or a negative sentiment. They have trained models with Naive Bayes, Maximum Entropy and SVM machine learning algorithms and came to the conclusion that these algorithms can perform well using the method they approached, reaching accuracies above 80%, with Naive Bayes showing better results [2]. Pak and Paroubek divided their tweet collection in three different groups of sentiments, the positive, the negative and the neutral emotions [1]. Similar to [2], giving importance on emoticons and training models with Naive Bayes classifier and SVM machine learning algorithms, with Naïve Bayes showing better results one more time. Later in 2011, [3] made an experiment in which they worked with Part-of-Speech, lexicon and micro-blogging features (the emoticons as the previous referred articles and other features to capture presence of sentiment). It was then concluded that the Part-of-Speech features may not be very useful for sentiment analysis in tweets due to the different language used among Twitter communities, with the remaining features proving to be more important to capture the presence of a sentiment in a tweet.

The pre-processing phase in the three articles was similar, with the use of tokenization to segment text, remotion or replacement of the external URLs with the token "URL", remotion of letters repeated more than two times in a row in a word, for example the words 'huuuungry' and 'huuungry' would be converted into the token 'huungry'.

III. METHODS

A. Data Description

Twitter[1] is an online news and social networking service where users communicate and interact with each other through short messages known as "Tweets". Tweets have a 280 character limit, and may include symbols such as emojis, which can be used to express an idea or an emotion and hashtags, which can be used by users to group posts together by topic or type.

For this experience, we have used a dataset that has a collection of 49921 tweets, which had a number identification, the tweet content and a classification that could have the values of "pos" and "neg", that identifies if a tweet has a positive or negative sentiment. The dataset had an initial amount of 41377 tweets labeled as positive and 8544 tweets labeled as negative.

B. Baseline

TextBlob[2] is a python toolkit that contain multiple features to aid developers with natural language processing tasks. In this subsection, TextBlob's sentiment analysis model was explored in order to create a baseline for future experiments. This model classifies sentences using a pre-trained model returning the polarity values with the range $[-1.0, 1.0]$. To define the baseline, this model was used to classify the polarity of tweets from the initial dataset, labeling a Tweet negative if the polarity is $[-1.0, 0.0[$ and positive if $[0.0, 1.0]$. The polarity value 0.0, which is considered neutral, for our work was classified as positive, due to the fact that our data-set had a high amount of tweets labeled as positive, thus reaching a **baseline accuracy percentage of 79%**

C. Data Pre-Processing Data Cleaning

In this stage, we decided to apply a vast number of Text Mining and NLP techniques, in order to pre-process and normalize the data-set and be able to compare future experiments and obtain better results.

1) **Lower Case Text:** One of the first steps in pre-processing is the conversion of all characters to lower case
2) **Tokenizer:** Tokenization was used to separate words from each tweet for later sentiment analysis tasks. It was used TweetTokenizer[3] from nltk python library as it has additional features that work better on our data-set
3) **HTML Handling:** External HTML were removed with the help of BeautifulSoup[4] from python library
4) **Symbols** Remotion of non-alphanumeric symbols in tweets
5) **Text processing** Lemmatization is the process of determining the canonical form of a word based on its intended meaning.
   Stemming is an algorithm to remove the suffix of words. For this we used ntlk's[5] WordNetLemmatizer and SnowBall Stemmer.
6) **Stopwords** Stopwords were removed with nltk[6] python library.

Table I, describes the experiments done with a variation with the pre-processment techniques listed above.

D. Sentiment Lexicon

Saif Mohammad and Peter Turney developed NCR Word-Emotion Association Lexicon, also known as EmoLex[4]. Emolex contains a list of 16494 English words and their associations with two sentiments (positive and negative) and eight emotions(fear, trust, surprise, sadness, joy, anger, antecipation and disgust). In this part of the experiment, the EmoLex Sentiment Lexicon was applied to the clean dataset to determine if a tweet either has a positive or a negative sentiment. To do so, we extracted the EmoLex list into a dictionary with the word and the associated value. The values are '-1' for words with a negative sentiment, '1' for words with a positive sentiment, and '0' for words that contain none or both positive and negative sentiments associated. Afterwards, the clean dataset was examined with the words from the dictionary and the sum of the values in each tweet would determine if a tweet has a positive or negative sentiment. A negative result from the sum would result in a negative sentiment, '0' and positive results would result in a positive sentiment. The '0' result was considered positive due to the high amount of tweets labeled as positive in the dataset. The accuracy obtained from this experience was 78.2%.

TABLE II
EMOLEX DATA-SET STATISTICS

Word Count	Negative	Positive	Neutral	Both (Pos/Neg)
14182	3243	2231	8627	81

[1]https://twitter.com
[2]http://textblob.readthedocs.io/en/dev/
[3]http://www.nltk.org/api/nltk.tokenize.html
[4]https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[5]http://www.nltk.org/api/nltk.stem.html
[6]https://www.nltk.org/book/ch02.html

TABLE I
DATA PRE-PROCESSING MODELS TESTED

Pre-Processing Model Nº	Lower Case	Tokenizer	Html Handling	Symbol Removing	Text Processing	Stop Words Removal
PPM.1	yes	nltk.TweetTokenizer()	BeautifulSoup(lxml')	yes	nltk.WordNetLemmatizer()	nltk.stopwords('English')
PPM.2	no	nltk.TweetTokenizer()	BeautifulSoup(lxml')	no	nltk.WordNetLemmatizer()	no
PPM.3	yes	nltk.TweetTokenizer()	no	no	nltk.SnowBallStemmer()	no
PPM.4	yes	nltk.TweetTokenizer()	no	no	nltk.SnowBallStemmer()	nltk.stopwords('English')

E. Vectorization

For this section we implemented our own version of Count Vectorizer, which created a new dictionary with all the words from the data-set and the respective number of occurrences, which can be found in the source code. Later we applied Tf-idf[7] from scikit learn with the Count Vectorizer dictionary to find out which words have more weight in the document.

F. Machine Learning Algorithms

Supervised machine learning algorithms are often used for text classification. The algorithms used in the experience were Logistic Regression, Support Vector Machine and Naive Bayes from Scikit learn[8].

1) ***Logistic Regression****:* Logistic Regression, also known Maximum Entropy, is described as a classifier that classifies an observation into one of two classes [5]. To do so, it measures the relationship between categorical features by calculating the probability of occurrence of particular outcomes with a natural logarithm function.

2) ***Support Vector Machine****:* Support Vector Machine (SVM) is mostly used for problems related with classification. SVM plots each feature in a n-dimensional space, where n represents the number of features and the value of each feature represents a particular coordinate. The classification is performed by finding the hyper-plane that separates the two classes.

3) ***Naive Bayes****:* Naive Bayes is an algorithm that is known for its good results in text classification, like SVM. It is based on the Bayes' theorem, which calculates the probability of an event occurring given that another event has occurred.

IV. Experiments

The following experiments were conducted between the Algorithms mentioned above and the data Pre-Processing Models detailed in the Table I.

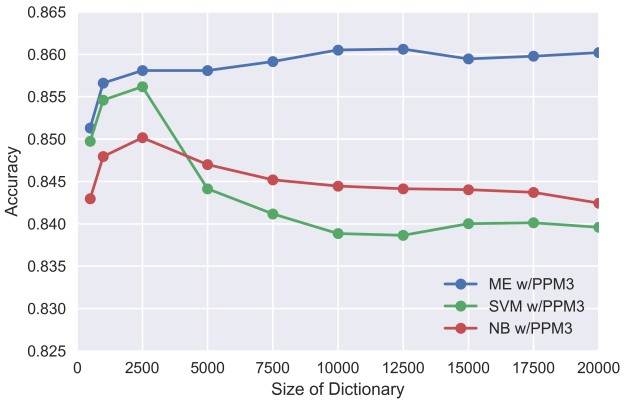

Fig. 1. Simulation with PPM3 pre-processing.

[7]http://scikit-learn.org/stable/modules/feature_extraction.html#loading-features-from-dicts

[8]http://scikit-learn.org/stable/

Our main experience was focused on discovering a correlation between a Dictionary Size Parameter, and the resulting Accuracy. Thus, we have limited our Count-Vectorizer and TF-IDF Features size from 500 to 20000 most used words.

It was chosen to plot the results with the pre-processing models PPM1 and PPM3 as they have a higher variation in terms of pre-processing, with PPM2 and PPM4 showing similar results.

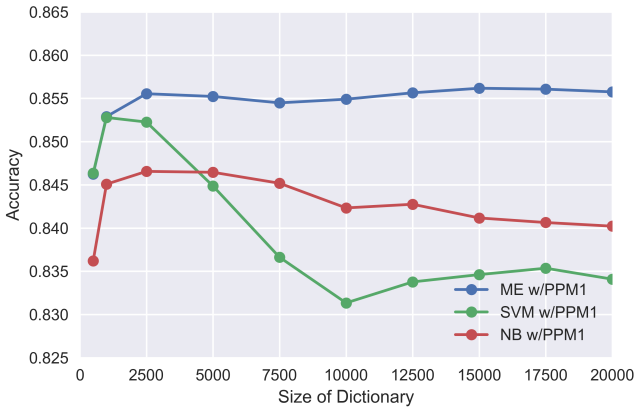

Fig. 2. Simulation with PPM1 pre-processing.

V. Results

The graphs in Fig. 1 and Fig. 2 show the accuracy performance of the pre-processing models PPM1 and PPM3, with PPM3 showing slightly better results.

The graph in Fig. 3 shows the highest percentage of accuracy for the different pre-processing models for each of the Machine Learning algorithms specified in the previous section, even thought the pre-processing may vary due to the Dictionary Size Parameter.

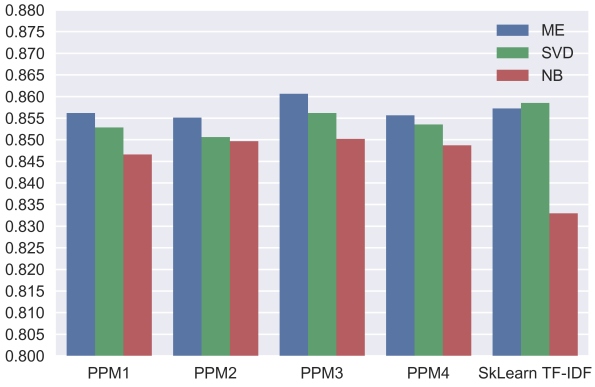

Fig. 3. .

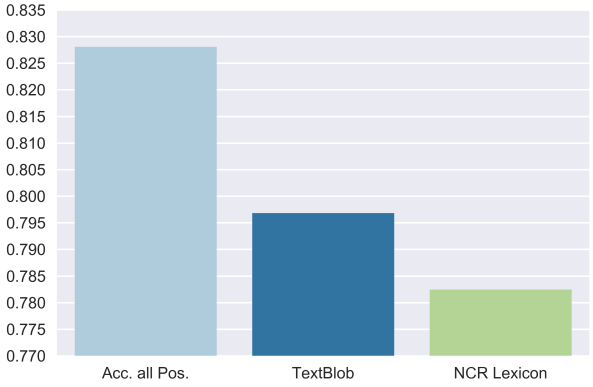

Fig. 4. .

As we can see, the Logistic Regression reached the highest accuracy, with a measure of 86% with the pre-processing model PPM3 and a Dictionary Size of 10000 words

The graph in Fig.4 shows a comparison of the different 'baselines' in our dataset, with NCR getting the lowest result, after TextBlob. 'Acc. all Pos.' is the accuracy we'd reach in case we pull all the classifications as positive, due to the high amount of positive labeled tweets in the data-set.

VI. Conclusions

After evaluating the results obtained, we noticed that Logistic Regression obtained better accuracy levels in most experiences, except in the Scikit learn TF-IDF experience. We can conclude that the remotion of StopWords was not a critical factor in the pre-processing phase, as we got better accuracy results in PPM3 pre-processing model than PPM4 with Logistic Regression.

VII. Statement

- Rafael Soares: 50%
- Tomás Machado: 50%

References

[1] Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of LREC.

[2] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report, Stanford.

[3] Kouloumpis, E., Wilson, T., & Moore, J. (2011). Twitter sentiment analysis: The Good the Bad and the OMG!. In Proceedings of the 5th International AAAI Conference on Weblogs and Social Media.

[4] Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. Computational Intelligence, 29(3):436–465.

[5] Speech and Language Processing. Daniel Jurafsky & James H. Martin.