Sentiment Analysis Using Supervised Machine Learning Models

Lobera del Castillo, José Luis
Machine Learning I
Universidad Panamericana
Aguascalientes, México
0224643@up.edu.mx

Abstract—**Sentiment analysis is a Machine Learning technique that detects the underlying sentiment of a sentence or text.**

The process classifies text as either positive, negative, or neutral. This learning techniques are used to evaluate a piece of text and determine the sentiment with which the sentence was uttered

Keywords—Machine Learning, Classification, Regression, Sentiment Analysis.

I. Introduction

Nowadays we live in a world where verbal communication it's not the only way to transmit sentiments. The first written letter was sent 500 years BC, the humans had to be precise and specific with their words to communicate the exact idea and sentiment. Then, at the beginning of the XIX century, morse code arrived, making even harder to transmit emotions with a limited number of words and characters, for example the famous "S.O.S" signal, using only three characters you could communicate an urgency. Centuries later, around 1985, SMSs arrived, **Short Message Services** gave the phone users a communication alternative, but this messages still had a limitation: you had to interpret the sentiments, feelings, and urgency behind them. Twelve years later, in 1997, the **emojis** arrived, an embedded pictogram of faces that could be included next to messages. This made easier to distinguish between emotions, e.g., "don't be late ☺" and "don´t be late ☹".

Natural Language Processing (NLP) is a subfield of Machine Learning concerned with the interactions between computers and human language, from syntactic analysis to relational semantics.

Sentiment Analysis or Opinion Mining is a subfield of NLP that studies and learns the attitude, sentiments, evaluations, attitudes, and emotions of a speaker/writer based on the computational treatment of subjectivity in a text.

Humans can almost always understand all the aspects behind a text without any trouble, so why is Machine Learning making efforts on understand it too?

The internet and social media are full of messages, comments, opinions, reviews, and many other types of text.

Imagine a content creator wants to know how his followers are expressing of him, or a shopping site wants to know how the customers feel about their products. They cannot read every comment or every review, and sometimes they do not care about the content of the text, but the sentiment or feeling behind them.

This is where **Sentiment Analysis** comes in, the Machine Learning algorithms can determine the exact percentage of positivity or negativity in each text allowing companies to make sense out of data by being able to automate this entire process.

Although the text comes from emails, chats, social media, surveys, articles, and other documents, programming languages help to automate these processes with their $scrapping^1$ tools.

Sentiment Analysis is also useful for practitioners and researchers, especially in fields like sociology, marketing, advertising, psychology, economics, and political science, which rely a lot on human-computer interaction data.

II. The Dataset

The starting point of this project was a Dataset uploaded to Kaggle, an online community of data scientists and machine learning practitioners, by M Yasser H, an Indian AI & ML Engineer.

The $\boldsymbol{Twitter}^{2}$ **Tweets Sentiment Dataset** is a **Comma Separated Values** (CSV) document, which contains only four columns: *textID, text, selected_text,* and *sentiment.*

[1]. *textID*: The twitter tweet identifier.

[2]. *text*: The main tweet text.

[3]. *selected_text*: The most significant part of the text, the words that could define the sentiment of it.

[4]. *sentiment*: **Positive**, **Negative** or **Neutral**.

The file contains 27481 registers from which 8582 are positive, 7782 negative and 11118 neutral tweets. Each register contains one of the three sentiment classification according to what the author thought the tweet was transmitting.

1. **Web Scrapping:** Techniques for extracting data from websites.
2. **Twitter**: American microblogging and social networking service on which users post and interact with messages known as "tweets".

III. Analysing the Data

When the file was first uploaded it looked like this:

ID	Twitter Tweets Data Frame		
	Text	***Selected Text***	***Sentiment***
549e99	"Sooo SAD I will miss you here in San Diego!!!"	"Sooo SAD"	Negative
16fab9	"I really really like the song Love Story by Taylor Swift"	"Like"	Positive
3c4a21	"Both of us I guess..."	"Both of us I guess..."	Neutral

From the four columns the dataset contains only two will play a part on the algorithm, the most important: the selected text, which contains the key words that give sense to the tweets; and of course, the tweet classification.

So now the data frame looks like this:

ID	Twitter Tweets Data Frame	
	Selected Text	***Sentiment***
1	"Sooo SAD"	Negative
2	"Like"	Positive
3	"Both of us I guess..."	Neutral

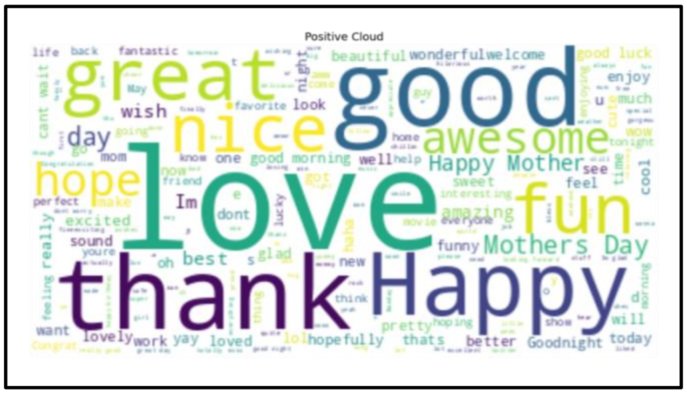
Now, with the key words selected, an analysis could be made to see the weight of the keywords, this means that the more a word appears on the "Selected Text" column, the more weight the sentiment will have.

To obtain these key words a "Count" or a "Top" method could have been performed, but to get a more visual insight, Text Clouds were generated, obtaining the following data. *(Figures a, b, c)*

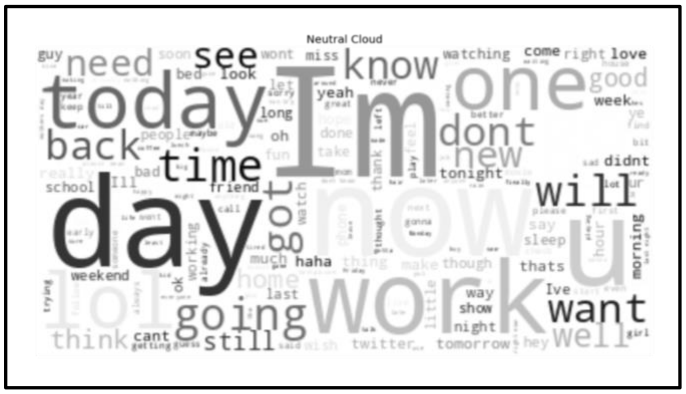
The Text Clouds suggest the following from the Data Frame selected text:

The most repeated words on positive tweets are words that have a positive meaning, words like **love**, **happy**, **fun**, **awesome**, are words that suggest positive emotions so having them in a sentence rises the probability of it being positive. The same logic applies to negative words like **sad**, **sorry**, **hate**, **bored**, **bad**, these words rise the possibility of a text being negative.

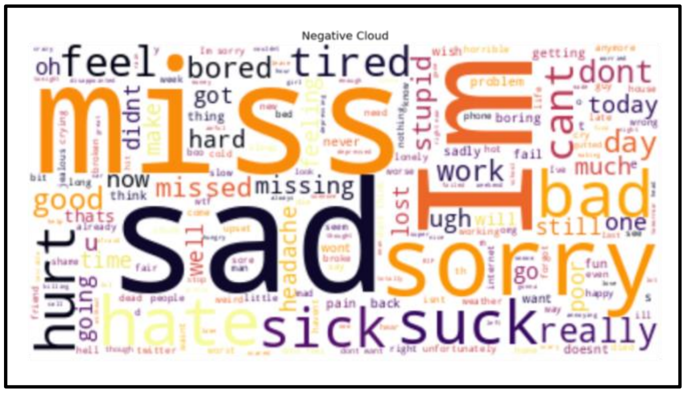
There are some worlds that do not affect the positive or negative probabilities, but sum to a neutral state where it does not affect neither sides, words like **today**, **time**, **work**, **can't**, **don't**, are words that can be used both in positive or negative sentences, e.g., "I **couldn't** be more happy" and "I **couldn't** stop crying", although "not" is an auxiliary verb to indicate negativity there are other words that have a higher weight when deciding a sentence sentiment.



[a] Positive Cloud Graph



[b] Neutral Cloud Graph



[c] Negative Cloud Graph

IV. Preprocessing

To obtain the best results once we train our Machine Learning models, the Data Frame must be adjusted for it to fit in the models. The following processes were made to the columns during the preprocess phase:

A. Text to Lower Case

All the characters are transformed into lower case. Although studies have shown that writing in Upper Case increments the intensity of the emotion, the sentiment is still the same and the Case is irrelevant, e.g., "**I can't do this anymore**" and "**I CAN'T DO THIS ANYMORE**" are both negative states, but the second one is more intense. Without transforming all characters, the model would detect "**Happy**", "**HAPPY**" and "**happy**" as different words.

B. *Remove punctuations*

Punctuations, like Cases, are irrelevant when talking about sentiment, it is true that exclamation marks add emotions to sentences and question marks change the pronunciation of it, but in the end, the sentiment remines. Other punctuation marks such as commas and points must be also removed because their purpose is no more than to mark the reading rhythm.

C. *TF-IDF String Vectorization*

Term Frequency-Inverse Document Frequency (TF-IDF) is a String Vectorization Method which calculates the importance of the word in the dataset. TF-IDF contains two concepts **Term Frequency** (TF) and **Inverse Document Frequency** (IDF).

Term Frequency is the frequency of a word in the Data Frame, it can be defined with then next formula:

$$TF = \frac{No.\ of\ times\ word\ apear\ in\ document}{Total\ number\ of\ words\ in\ the\ document}$$

Most of the times, the number of appearances of the words will be very large, to reduce the value of frequency count, the algorithm adds a logarithmic term to damp the performance of IDF and to reduce the frequency count of TF.

$$w_{t,d} = \begin{cases} 1 + log_{10} TF_{t,d} & if\ TF_{t,d} > 0 \\ 0 & otherwise \end{cases}$$

Inverse Document Frequency is another concept for finding out the relevance of a word. It is since less frequent words are more informative and important. IDF is represented with the next formula:

$$IDF = log_{10} \frac{Number\ of\ samples}{Number\ of\ samples\ where\ the\ word\ apear}$$

So, **TF-IDF** is the multiplication between TF and IDF. It reduces values of common words that are used in different samples.

V. Machine Learning Models

Two supervised learning models for classification learned from the Data Frame obtaining similar scores, the first one: Naïve Bayes, and the second one: Logistic Regression.

Both models were tested with and without cross-validation and received the same fitting sets. The Data Frame was divided 80% as training set and 20% as testing set.

A. *Naïve Bayes Algorithm*

The first algorithm that learned from the set was the Naïve Bayes Model, a probabilistic classifier. The model achieved a score of 81.73% and a macro F1 score of 81.36%.

A confusion matrix was calculated and obtained the following results:

Category	**Confusion Matrix**		
	Negative	***Neutral***	***Positive***
Negative	582	86	91
Neutral	138	953	108
Positive	33	46	711

A Classification Report was also logged with the following metrics:

Metric	**Classification Report**			
	Precision	***Recall***	***F1-Score***	***Support***
Negative	0.77	0.77	0.77	759
Neutral	0.88	0.79	0.83	1199
Positive	0.78	0.90	0.84	790
Accuracy			0.82	2748
Macro avg	0.81	0.82	0.81	2748
Weighted avg	0.82	0.82	0.82	2748

Then, a Cross Validation technique was performed to find the training set that best fitted the model, the best Cross Validation model achieved an 82.07%, 0.71% higher than the original model.

B. *Logistic Regression*

The second algorithm to learn from the tweets set was the Logistic Regressin model, an statistic regression classifier.

The logistic model achieved a score of 82.64% and a macro F1 score of 82.41%.

Its confusion matrix indicated the following performance:

Category	**Confusion Matrix**		
	Negative	***Neutral***	***Positive***
Negative	615	91	93
Neutral	117	940	101
Positive	21	54	716

And the Classification Report stated the following:

Metric	**Classification Report**			
	Precision	***Recall***	***F1-Score***	***Support***
Negative	0.82	0.77	0.79	799
Neutral	0.87	0.81	0.84	1158
Positive	0.79	0.91	0.84	791
Accuracy			0.83	2748
Macro avg	0.82	0.83	0.82	2748
Weighted avg	0.83	0.83	0.83	2748

The same Cross Validation technique was aplied to improve the Logistic Regression performance, the score increment was higher than the Naïve Bayes one, with an 0.91% increment and a total F1 performance of 83.33%.

C. Comparisson Between Models

Both models had great performances, eight out of ten sentences are correctly classified. The metrics indicate a very similar performance, even though the Logistic Regression is 0.91% more accurate. The strongest advantage of Logistic Regression can be observed in the **Negative** class precision, from all the Logistic Regression **Negative** predictions, 82% were negative, 5% higher than the Naïve Bayes **Negative** predictions.

D. Positive and Negative Probabilities

Apart from the prediction, both models calculate the probability of an input being positive, negative, or neutral as shown below:

Model	**Input text:** "What a boring game omg..."		
	Negative	***Neutral***	***Positive***
Naïve Bayes	**72.388%**	20.678%	6.935%
Logistic Regression	**65.928%**	27.249%	6.823%

Model	**Input text:** "Today is Friday 13"		
	Negative	***Neutral***	***Positive***
Naïve Bayes	25.989%	**60.298%**	13.713%
Logistic Regression	30.304%	**60.349%**	9.347%

Model	**Input text:** "I love this stupid show!!!"		
	Negative	***Neutral***	***Positive***
Naïve Bayes	33.576%	19.062%	**47.361%**
Logistic Regression	16.277%	5.282%	**78.441%**

VI. OTHER APROACHES

There are hundreds of ways to Analyze Sentiment, and another hundreds of data sets the models can learn from.

From the best models for Sentiment Analyze we find, apart from Naïve Bayes and Logistic Regression, the *Support Vector Machine* (SVM), *Recurrent Neural Network* (RNN), *Convolutional Neural Network* (CNN), *Random Forest*, and *Long Short-Term Memory* (LSTM).

Another approach is the VADER Sentiment Analysis, VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media.

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. The big difference VADER has against the previous algorithms is that VADER supports and understands emojis, Slangs, and Emoticons.

VII. CONCLUSION AND IMPROVEMENT PROPOSAL

The Sentiment Analysis showed great results, even though their prediction scores were under 90%, both model performances can be high rated, considering that sometimes even humans can´t understand each other.

Sentiment analysis has a big field in computer science, most apps and services are constantly being reviewed by users, and even though most reviews are collected in categorical scales, there is a great number of users that expressing their opinions in the comment sections.

As mentioned before, there are tons of ways to approach Sentiment Analysis, so as an improvement proposal, I would. Train more Machine Learning Algorithms and compare their performances. Use different data sets to prevent overfitting would also be a proposal and getting into Natural Language Processing and Text Preprocessing algorithms would great tools and would allow a bigger field of view on text processing.

ACKNOWLEDGMENT

To Dr. Claudia Nallely Sánchez Gómez for everything you have taught us during these 3 years, for your infinite support and guidance. Thank you.

REFERENCES

[1] Liu B. (2015). Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. United Stated: Cambrige University Press

[2] VanderPlas J. 2016. Python Data Science Handbook. United States: O'Reilly Media, Inc.

[3] Medium. (2018). Simplifying Sentiment Analysis using VADER in Python (on Social Media Text). June 1st 2022 at: https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52

[4] Monkey Learn. (2022). Sentiment Analysis: A Definitive Guide. June 1st 2022 at: https://monkeylearn.com/sentiment-analysis/

[5] BRAND24. (2022). What is Sentiment Analysis? An Ultimate Guide for 2022. June 1st 2022 at: https://brand24.com/blog/sentiment-analysis/

[6] Medium. (2018). Natural Language Processing: Text Data Vectorization. June 1st 2022 at: https://medium.com/@paritosh_30025/natural-language-processing-text-data-vectorization-af2520529cf7

[7] Medium. (2019). TF IDF | TFIDF Python Example. June 1st 2022 at: https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76