Project - Sentiment Analysis

Laura Arias Fernandez

May 2, 2022

Abstract

Sentiment analysis is a sub-field of Natural Language Processing (NLP) that focuses on the categorization of subjective opinion of the text (positive, negative, or neutral). There are two main components to sentiment analysis in natural language processing: feature extraction and the machine learning model. In this paper, we examine sentiment analysis on Twitter data. Specifically, we explore the use of different feature extraction techniques (3-grams, 4-grams, bag of words, and tf-idf) and different machine learning models (naive Bayes classifiers, SVM, logistic regression) to compare the model accuracy. The tf-idf feature extraction technique in conjunction with logistic regression was shown to have the best performance.

1 Introduction

Natural Language Processing (NLP) is an area of computer science, which focuses on developing the computer's understanding of text. Sentiment analysis is a form of Natural Language Processing that consists of analyzing texts to infer the given subjective opinion (positive, negative, or neutral) on a particular topic. Nowadays, corporations are using this technology to evaluate how consumers are reacting to products, for example television companies use it to analyze how much their viewers liked a movie, but the applications can go far beyond their current use.

In the modern world, mental illnesses, like depression, have become a new normal. According to the World Health Organization (WHO), approximately 280 million people worldwide suffer from depression[14]. Some poeple are so depressed that they reach a point where they feel so helpless that they decide to comit suicide. The response to combat such a mental health pandemic is extremely complex and any small contribution can make a huge difference.

People today use social media as an emotional outlet, where they can express emotions, so being able to classify their statements into positive and negative can be of significant help. Something as subtle as being able to predict if a comment a person made on twitter was positive or negative can help determine whether that person may need help, and proceed to offer him/her mental health resources.

Twitter is a social media application where users share their personal statements in 140 characters or less; such statements are called "tweets". Although other social media platforms, like Meta, also allow individuals to express themselves, most people use those platforms to connect with their family and friends, and Twitter to share their personal thoughts[8]. Hence, if needed to analyze individuals thoughts, Twitter would be the more reliable in that regard.

This paper aims to compare and contrast various algorithms and feature extraction techniques used for sentiment analysis. Specifically, I focus on 3 different algorithms: Naive Bayesian classifiers, SVM, and Logistic Regression. For each of these methods, four different feature extraction techniques are used on the dataset: tf-idf, bags of words, and 3 4 n-gram. Concretely, these techniques are applied to the Sentiment140 dataset[11] that includes twitter posts gathered by the twitter api, containing approximately 1.6 million tweets, each labeled with a value of 0 if the tweet is considered negative and 4 if the tweet is considered positive. In order to provide accurate tests, I decided to use the common split ratio 70:30, in which 70% of data will be used for training and 30 % for testing.

In this paper, related work on sentiment analysis is reviewed in section 2, the different feature extractions techniques and machine learning algorithms used in the experiment are discussed in depth section 3, and the experiment design and software packages used are explained in section 4. Moreover, the results and discussion of the experiment can be found in section 5 and 6 respectively. Finally the paper concludes in section 7 with a summary of results and future possible ameliorations of the work proposed.

2 Related Work

2.1 Algorithms & Feature Extraction

A.Vishal and Sonawane[3], in their research paper, describe multiple methods for performing sentiment analysis on a document. The first method described is "Lexicon based", which consists of matching every word on the text with a dictionary holding how positive or negative such a word is, thereby obtaining the polarity of the text. Lexicon Based Sentiment Analysis can provide fast results and does not require labels to perform such inference. However, the main disadvantage of this method is that having more positive words than negative words does not necessarily imply the text is positive[10]. For instance, "I loved the food, but hated everyone" would be labeled as neutral despite its clear negative sentiment. Another approach mentioned by the authors was Cross-lingual Sentiment Analysis (CLSA), which has been found to be very useful when performing sentiment analysis on languages with very limited data, such as Catalan (a language spoken in a region of Spain). This approach consists of using the data of well documented languages, such as English or Spanish to perform sentiment analysis, and use such inferences to perform sentiment analysis on the text written in the less popular language[6]. Overall, the most common and well-known approach to perform sentiment analysis is supervised learning (Machine Learning), which utilizes labeled data containing the text and the polarity of the text (positive or negative) to infer a function that will then be used to determine the polarity of unlabeled texts. The machine learning algorithms used for sentiment analysis in the paper by A.Vishal and Sonawane[3] were svm and naive bayes and maximum entropy, two of which (svm and naive bayes) are explained in depth in section 3 of this paper as they are implemented.

Unfortunately, we cannot feed the sentences as a sequence of characters into the machine learning algorithm. Instead, we need to find a way to map sentences to numbers. This process of mapping text data into numbers is called feature extraction, for which there are multiple techniques that can be used. In A.Vishal and Sonawane's[3] research paper, only the n-gram feature extraction technique is explored with n being 1,2, and 3. This feature extraction technique is explained in depth in the following section.

Novikova and Stupnikov's[12] research paper focuses on comparing five different machine learning sentiment classification algorithms (svm, logistic regression, random forest classifier, gradient boosting classifier, kneighbors classifier and multinomial naive bayes) using tf-idf technique as a method for feature extraction. This technique's full name is term frequency-inverse document frequency (tf-idf), and consists of counting the number of occurrences of each word in a text and multiplying by the logarithm of the number of documents divided by the number documents that contain that word[23]:

for a term i in document j:

$$w_{i,j} = tf_{i,j} * log(N/df_i)$$

$tf_{i,j}$ = number of occurrences of i in j
$df_{i,j}$ = number of documents containing of i
N = total number of documents

This method gives a numeric representation to each word, explaining how much information that word provides. If a word is present in most text, it does not have a lot of weight since it means such word does not give any polarity relevant information about the text. On the other hand, if the word is not common in most documents, but has high occurrence in a given text, its numerical value is high,

implying that it gives important information about the polarity of such text.

The training data fed by Novikova and Stupnikov's[12] into the tf-idf feature extraction to create numeric data to feed into each of the five ml algorithms mentioned above (svm, logistic regression, random forest, gradient boosting, k-neighbors classifier, multinomial naive bayes), consists of 10000 social media posts from the VKontakte social network (6380, 2700 and 600 posts of neutral, positive and negative classes respectively). Once the training finished, the results were reflected based on the f1-score, and were 0.70, 0.68, 0.66, 0.66, 0.59 and 0.61 respectively if 3 classes classification (positive, neutral and negative) were considered, and 0.48, 0.40, 0.45, 0.35, 0.20, 0.20 respectively if only positive and negative classes were considered.

Similar research to the papers mentioned above was also conducted by Psomakelis [17]. Nevertheless, in this paper there were 3 different feature extraction methods compared (n-gram, bag of words, and n-gram graphs) in order to predict the sentiment analysis in their tweets dataset. In the research, the bag of words feature extraction technique produced inaccurate results, regardless of the algorithm used. The best result gathered using this feature extraction technique was through the use of the machine learning algorithm Naïve Bayesian, obtaining an accuracy of 45.07%. In the case of the n-gram feature extraction method, regardless of the n, logistic regression worked best. The accuracy results using this feature extraction technique and the logistic regression ml algorithm were 52.19%, 65.21%, and 75.88% for the 3, 4 and 5-Grams feature extraction techniques respectively. Overall, in this experiment, the accuracy was much higher when the n-gram graph technique was used for feature extraction. This technique is very similar to the n-gram technique, but in this case, the interaction/relation that each n-gram has with its neighbors is captured. The accuracy results of this feature extraction were 83.15%, and 94.52% on the 5 and 4-Gram Graphs respectively, both through logistic regression.

2.2 Evaluation Scores

In terms of evaluation scores used for sentiment analysis, there are four main ones: accuracy, precision, recall and f1[3]. In the case of accuracy, it only measures the percentage of examples that were predicted correctly, while precision is the fraction of positive predictions that are actually positive, and recall is the fraction of positive examples that were actually predicted as positive. Due to the fact that both false positives and false negatives are crucial to understand the performance of the algorithm, both precision and recall need to be taken into account, something the f1-score does through the following formula[1]:

$$F1 = 2 * \frac{precision * recall}{precision + recall} = \frac{truepositives}{truepositives + 0.5(falsepositive + falsenegatives)}$$

Although different research papers used different evaluation scores ([3] [12] uses f1-score and [17] uses accuracy), F1 score is a much better representation of the results than the accuracy, since it is a weighted average of precision and recall and is specifically useful if your testing data has more examples of one class than the other[22].

2.3 Data Cleaning

Moreover, before any of the algorithms or feature extractions are implemented, it is crucial to clean the data. This step is key in classification because, when downloading the data, it can be formatted incorrectly or be filled with unnecessary characters such as punctuations and stopwords. Such unclean data can have very negative effects in the training leading to inconsequential insights of data[19]. A.Vishal and Sonawane[3] describe the process of data cleaning, including the removal of all urls, punctuations, presence of acronyms, and correction of spelling. Specifically, they mention the importance of removing stop words like "but, a, and, with" in the data cleaning process, since these words do not contribute to the sentimental analysis classification and, due to their common presence, they can negatively affect the classification. Additionally, Project Pro website[2] takes data cleaning to the next

level by explaining another technique: Stemming and Lemmatization. This consists of transforming every single word into their root; for instance, go and going should both be represented as the word go. This technique plays a crucial role in feature extraction for techniques like bag of words, as if we implement this cleaning method, go and going will be counted by the feature extraction technique as the same word occurring twice, rather than as two different words.

Another important step in data cleaning is the removal of emojis, which would actually fall under the removal of punctuation, since a smiley face is represented with a colon and a parenthesis - ":)". Although emojis can be useful when it comes to identifying sentiment analysis of a text, most papers [3] [12] [17] decide to remove them as it makes data processing much easier. Nevertheless, Novikova and Stupnikov[12] reserve a section in their research paper to compare logistic regression sentiment analysis with and without emoticons and it shows that leaving them improves the f1-score by 2% from 0.68 to 0.70.

My work aims to compare and contrast three different machine learning algorithms used in the papers mentioned above: Naive Bayesian classifiers, SVM and Logistic Regression, alongside four different feature extraction techniques mentioned, in at least, one of the papers above: bags of words, tf-idf, and 3 and 4 n-gram.

3 Approach

As described above, three different machine algorithms are going to be trained with the labeled data found in kaggle, Sentiment140 dataset[11], which include approximately 1.6 million tweets, each labeled with a value of 0 if the tweet is considered negative or 4 if the tweet is considered positive.

3.1 Machine Leaning Algorithms

3.1.1 Naive Bayesian Classifier

The first algorithm to be implemented is the Naive Bayesian classifier, which consists of calculating the probability that a given tweet is of a given polarity based on the probability that each of the words of the tweet is present in a message of such polarity, then multiplying each of these probabilities together. In training, this algorithm will go through all the positive tweets and calculate the probability that a certain word appears in a positive tweet ($P(word \mid tweet = positive)$). Such probability is found by adding all occurrences of that word among the positive tweets and dividing it by the total amount of words in the positive tweets. This process will be performed during training for all the words in the vector, and repeated for all the negative tweets, thereby obtaining $P(word \mid tweet = negative)$. Consequently, the Bayes theorem can be used to obtain:

$$P(+tweet \mid text\ data) = P(+tweet) * \prod_{i=word}^{textdata} P(word \mid +tweet)$$

$$\textbf{where } P(+tweet \mid word) = \alpha\ P(word \mid +tweet) * P(+tweet)$$

$$P(-tweet \mid text\ data) = P(-tweet) * \prod_{i=word}^{textdata} P(word \mid -tweet)$$

$$\textbf{where } P(-tweet \mid word) = \alpha\ P(word \mid -tweet) * P(-tweet)$$

The probability of being a positive or a negative tweet mentioned in the formulas above can be calculated by counting the number of tweets of such polarity and dividing it by the number of total

number of tweets used in training. Overall, for this machine learning algorithm given an unlabeled tweet, it can be classify as positive if $P(tweet = positive \mid text) > P(tweet = negative \mid text)$, otherwise it is classified as a negative tweet.

3.1.2 Support Vector Machine

The second type of machine learning algorithm is SVM; Unlike naive bayes algorithm which focuses on maximizing the probability that the labeled tweets belong to that class, this one focuses on finding a hyperplane with the largest possible margin that maximizes the division between the two classes.

In the simplest possible situation, the classes are linearly separable, and consequently the algorithm will look directly at the labeled training data and find a hyperplane that has the greatest margin between both classes, thereby it only focuses on the training data closest to the hyperplane, called support vectors. Nevertheless, the classes may not be linearly separable; In this case the SVM can be modified to implement a technique called the kernel trick, which is a function that transforms the data points into a higher dimension where the data is linearly separable, and for which the SVM can find a hyperplane[20]. There are multiple kernels available, such as the gaussian kernel or the polynomial kernel[21]. Without testing how each kernel performs it cannot be infer which kernel to choose, yet for this experiment the linear kernel + SVM will be compared to the other algorithms, since the related work[3] [12] used such kernel in their algorithm and thereby in order to contrast with such work, this project focus exclusively on the linear kernel.

3.1.3 Logistic Regression

The third and final machine learning algorithm is logistic regression, which is a probabilistic classifier similar to linear regression, but for this algorithm instead of trying to fit a linear function through the data, an s shape function is fit; one that outputs a value between 0 or 1. The goal of logistic regression training is to find a function that for positive tweets outputs a value closest to 1 and for negative tweets it outputs a value closest to zero[7]. This process is accomplished through gradient descent; During training logistic regression tries to find the parameters of the function that maximizes the likelihood of the samples to belong to their labeled class, thereby tries to find parameters that minimize the following cross entropy error[9]:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \quad \begin{cases} Cost(h_\theta(x^{(i)}), y^{(i)}) = -log(h_\theta(x)) & if \quad y = 1 \\ Cost(h_\theta(x^{(i)}), y^{(i)}) = -log(1 - h_\theta(x)) & if \quad y = 0 \end{cases}$$

3.2 Feature Extraction

3.2.1 Bag of Words

In order to use text data in any of the machine learning models described above, the cleaned text data needs to be converted into numerical vectors. These vectors can be created using a technique called bag-of-words, which is the simplest method of encoding each word of a text. In this case, there is a vector which contains a space for each word that appears in at least one of the training data texts. For each of the texts, the encoding is as follows: there is a zero in every single position of the vector mentioned, except for the position within the vector for the words the text has; in this case, the vector has a value representing the number of occurrence of that word in the tweet[18]. For instance, imagine we had 3 texts “hello you,” “you are”, and “you you”. The general vector will have 3 spaces each representing 1 word [hello, you, are]. Because the first text has the words hello and you, each occurring only once in the text, it will be represented by [1,1,0]. Consequently, the other 2 texts will be represented by [0,1,1], [0,2,0].

3.2.2 TF-IDF

A drawback of the bag-of-words feature extraction technique is that there may be a word that is very frequently used in all the documents but does not provide a lot of information about the polarity/sentiment of the document, given that it does not allow to draw a distinction between the different texts in the data. On the other hand, Term Frequency - Inverse Document Frequency(tf-idf) vectorization technique solves this problem. Its process and formula is mentioned in section 2 of this paper, and it relies on a representation of each word based on its frequency in a text, and the inverse frequency of its occurrence in all the documents. In other words, a word has a high vectorized value if it does not appear very commonly in all texts but it has significant frequency on a given text.

3.2.3 N-gram

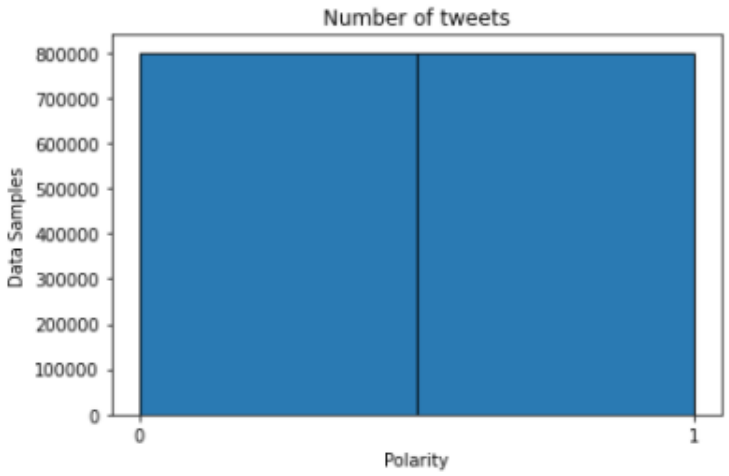
In both of the feature extraction techniques mentioned above, the words are considered separately, implying that there is no relationship between them; Nevertheless, in most cases it is very unlikely that words are better off separately and considering sequences of n-words rather than one word at the time may allow the feature extraction technique to capture more contexts around each word[13]. N-gram is a feature extraction technique that follows exactly the same process of vectorization as bag-of-words, but instead of considering one word for each vector space, it considers the sequence of n-words.

3.3 Final Steps

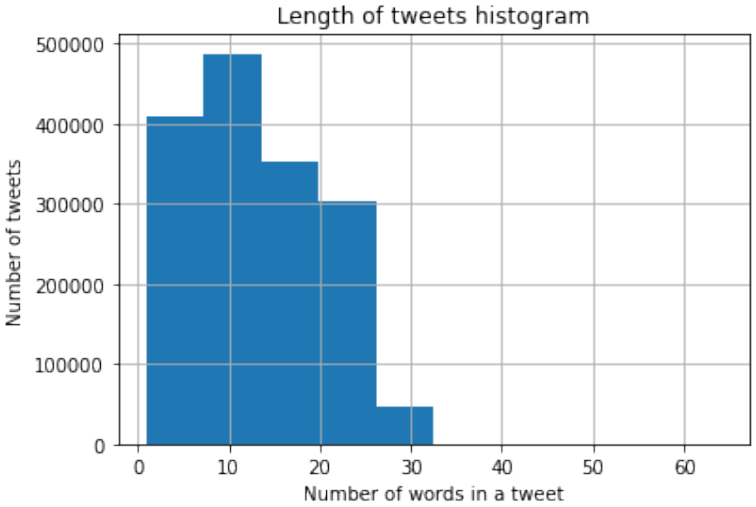
Once we have the dataset cleaned and mapped to real numbers using the different feature extraction techniques, we can feed that data into the different machine learning algorithms with 70% of the total dataset. The other 30% will be reserved for testing purposes since in order to test the accuracy of the algorithms the feature extraction technique, we need a percentage of dataset not used in training and for which we can use to calculate the f1-score. In section 2, it was mentioned why f1-score was a better predictor of how well the algorithm worked in more depth, but overall it was because it is a weighted average of precision and recall, meaning it takes both false positive and true negatives into account and unlike the accuracy score its result is not affected by the imbalances in number of samples for each class[22].

4 Experiment Design and Results

For the experiment I used the dataset Sentiment 140 found in Kaggle[11], which includes 1.6 million twitter posts, each labeled with a value of 0 if the tweet is considered negative and 4 if the tweet is considered positive.



(a) Tweets by Polarity



(b) Tweets by length

Figure 1: Visualization of Data

The first step after loading the dataset into google codelab was to perform data cleaning. Instead of writing my own code, I took advantage of Texhero[5], a python package which is built on top of pandas[15], and through which steemization, removal of stopwords, and any other data cleaning process explained in previous sections was able to be implemented in a few lines.

Although all the algorithms and feature extraction techniques can be implemented from scratch, following the explanation in section 3, I utilize an open-source library called scikit-learn[16]. The collection of raw tweets were passed to the feature extraction methods (TfidfVectorizer and CountVectorizer) in order to obtain a matrix representation of the tweets. Afterwards, such matrices were splitted into a 70-30 ratio, meaning that 70% of the tweets were used to train each of the models and 30% to test the precision and recall of the model.

Once data was splitted into training and testing groups for each feature extraction (tf-idf, bags of words, 3 and 4 grams), the training data for each matrix was passed to the machine learning algorithms (linear svm, logistic regression, naive bayes) in order to train the model. Finally the models were used to predict the test data subset. The actual labels of the test data were compared with the predicted values, and the f1-score for each model was calculated. In total 9 models were created; Table 1 shows the respective f1-score for each of the models.

	ALGORITHMS		
FEATURE EXTRACTION	**Multinonial Naive Bayes**	**Linear SVM**	**Logistic Regression**
Tf-idf	0.748	0.774	0.784
Bag of Words	0.756	0.765	0.779
3-gram	0.435	0.441	0.445
4-gram	0.141	0.145	0.145

Table 1: f1-score for each of the models

The full code of my experiment can be found in my google codelab notebooks, which I have uploaded to github[4].

5 Analysis

Given the results illustrated in Table 1 for the 9 different models, this section aims to expand upon those results using the knowledge of how each algorithm works as explained in section 3, in order to better understand what may be occurring during training and testing that resulted in the different performances.

Based on the results obtained, tf-idf feature extraction in combination with Logistic Regression as the machine learning algorithm resulted in the best model, obtaining a 0.784 f1-score. On the other hand, the model with the worst performance was 4-gram feature extraction with Multinomial Naive Bayes as it resulted in an f1-score of 0.141.

Specifically, the f1-score for Logistic Regression performed the best out of the 3 machine learning models regardless of the feature extraction used, which does in fact support the research conducted by Psomakelis[17] mentioned in section 2, in which for most of the feature extraction techniques, the model implementing logistic regression was shown to have the highest performance.

Nevertheless, my results do contradict Psomakelis' results regarding the f1-score when n-gram is used as a feature extraction. In my research the performance of the model decreased as the n-value

increased. On the other hand, Psomakelis' model resulted in a significant accuracy increase as the n increased, meaning the accuracy of 5-gram was much higher than the accuracy of the 4gram, which consequently was much higher than the accuracy of the 3-gram. My results may seem counterintuitive, yet there are multiple reasons why this may have occurred, but the one that makes the most sense is that the average length of words in my dataset is 14 as demonstrated in Figure 1b, meaning the text inputs are very small in size in order to consider very large groups of words. Further research should be conducted to accurately conclude the reason behind my results.

The models' performance in this experiment were very similar to the models constructed in Novikova and Stupnikov's[12] research, who compared svm, logistic regression, random forest classifier, gradient boosting classifier, kneighbors classifier and multinomial naive bayes using tf-idf as the feature extraction technique. In their research, Novikova and Stupnikov, found that the worst model for tf-idf from svm, logistic regression and naive bayes was naive bayes, which is also true for this paper's experiment. The only difference between their results and this experiment's results is their case the f1-score as reported in section 2 were much lower than the scores obtained here. The low f1-score in Novikova and Stupnikov's research may be due to the fact that their dataset contained only 10000 social media post, which results in a very low training dataset in comparison to the sentiment 140 dataset[11] used here to trained my model, who has a total of 1.6 million tweets.

Overall, the analysis of the models shows a slight preference for logistic regression as all the feature extraction techniques have a higher classification performance with that machine learning algorithm. Although results were mostly consistent with the related work found for sentiment analysis, some discrepancies were found, specifically relating to the n-gram technique, and therefore in order to accurately predict why that is, more experimentation needs to be performed.

6 Conclusion and Future Work

In this paper we have described several procedures using different machine learning algorithms for determining the sentiment (positive or negative) of tweets. We also explored how the different feature extractions techniques affected these machine learning algorithms. Our goal was ultimately to find the best combination of algorithm and machine learning technique to classify the sentiment of a tweet. Our results indicate that the logistic regression algorithm improves performance noticeably regardless of the feature extraction, which agree with the results of Psomakelis[17]. Our results also support Novikova and Stupnikov's[12] research by proving tf-idf performance compliments svm and logistic regression best. Specifically, the most effective model in our experiment was found to be the combination of logistic regression with tf-idf as a feature extraction.

With an n-gram as a feature extraction the bigger the n was the less effective our model became regardless of the machine learning algorithm, which contradicts the research conducted by Psomakelis[17] regarding that feature extraction technique. It might be beneficial to experiment with other types of text of longer lengths to understand why the opposite results occur. Moreover, another feature technique that could be tested is n-gram graphs. Comparing this feature technique with the n-gram technique could also allow us to test whether or not the assumption made by n-grams that each n-gram is independent from the others is true or not, and learn whether that assumption is what is causing the inaccuracy for that feature extraction technique.

Another technique to explore and see if it improves our model would be to allow emojis. Since emojis are a crucial part of the emotional content of a text, it can improve our f1-score; One way could be explored is By replacing the emojis for words that mean the same thing. Overall, these are just a few of techniques and methods that could be considered in future work to test if it improves sentiment analysis models performance.

References

[1] F-score, May 2019.

[2] 10 NLP techniques every data scientist should know, Feb 2022.

[3] Vishal A. and S.S. Sonawane. Sentiment analysis of twitter data: A survey of techniques. *International Journal of Computer Applications*, 139(11):5–15, Apr 2016.

[4] Laura Arias Fernandez. Lauraariasfdez/sentimentanalysis.

[5] Jonathan Besomi, Apr 2020.

[6] Sayali Borkar and Pushpak Bhattacharyya. Cross-lingual sentiment analysis.

[7] Gustavo Chavez. Understanding logistic regression step by step, Dec 2019.

[8] Caroline Forsey. Twitter, facebook, or instagram? which platform(s) you should be on, Jul 2021.

[9] Brendan Fortuner. Logistic regression.

[10] S. Kannan, S. Karuppusamy, A. Nedunchezhian, P. Venkateshan, P. Wang, N. Bojja, and A. Kejariwal. Chapter 3 - big data analytics for social media. In Rajkumar Buyya, Rodrigo N. Calheiros, and Amir Vahid Dastjerdi, editors, *Big Data*, pages 63–94. Morgan Kaufmann, 2016.

[11] KazAnova. Sentiment140 dataset with 1.6 million tweets, Sep 2017.

[12] Anastasia Novikova and Sergey A. Stupnikov. Sentiment analysis of short texts from social networks using sentiment dictionaries and blending of machine learning algorithms. In *AIST*, 2018.

[13] University of Cincinnati. Creating text features with bag-of-words, n-grams, parts-of-speach and more.

[14] World Health Organization. Depression, Sep 2021.

[15] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] Evangelos Psomakelis, Konstantinos Tserpes, Dimosthenis Anagnostopoulos, and Theodora Varvarigou. Comparing methods for twitter sentiment analysis. *arXiv preprint arXiv:1505.02973*, 2015.

[18] Purva. Quick introduction to bag-of-words (bow) and tf-idf for creating features from text, Dec 2020.

[19] Awan Rahman. What is data cleaning? how to process data for analytics and machine learning modeling?, Jun 2021.

[20] Sunil Ray. SVM | support vector machine algorithm in machine learning, Sep 2017.

[21] Shipra Saxena. SVM | support vector machine | how does SVM work, Mar 2021.

[22] Exsilio Solutions and * Name. Accuracy, precision, recall amp; f1 score: Interpretation of performance measures, Nov 2016.

[23] Haitian Wei. NLP pipeline 101 with basic code example-feature extraction, Apr 2019.