Optimizing Natural Language Processing Applications for Sentiment Analysis

Anderson Claiton Lopes[a], Vitoria Zanon Gomes[b] and Geraldo Francisco Donegá Zafalon[c]

Department of Computer Science and Statistics, Universidade Estadual Paulista (UNESP), Rua Cristóvão Colombo, 2265, Jardim Nazareth, São José do Rio Preto - SP, 15054-000, Brazil

Keywords: Natural Language Processing, Sentiment Analysis, Machine Learning.

Abstract: Recent technological advances have stimulated the exponential growth of social network data, driving an increase in research into sentiment analysis. Thus, studies exploring the intersection of Natural Language Processing and social network analysis are playing an important role, specially those one focused on heuristic approaches and the integration of algorithms with machine learning. This work centers on the application of sentiment analysis techniques, employing algorithms such as Logistic Regression and Support Vector Machines. The analyses were performed on datasets comprising 5,000 and 10,000 tweets, and our findings reveal the efficient performance of Logistic Regression in comparison with other approach. Logistc Regression improved the performed in almost all measures, with emphasis to accuracy, recall and F1-Score.

1 INTRODUCTION

Natural Language Processing (NLP) is an area of Artificial Intelligence and Linguistics dedicated to making computers understand statements or words written in human languages. Natural language NLP emerged to facilitate the user's work and to satisfy the desire to communicate with the computer in natural language (Khurana et al., 2022).

To perform NLP consistently, it is necessary to previously establish criteria to be followed, which led to the use of algorithms and tools for this problem. The first algorithm that stood out in history was the Georgetown–IBM experiment, in 1954 (Hutchins, 2004). Between 1964 and 1966, (Weizenbaum, 1966) developed the ELIZA program, considered the first chatbot. The big breakthrough in NLP came in the 1980s, thanks to advances in both hardware and software with the use of Machine Learning (Bonaccorso, 2017).

Sentiment Analysis is an area of NLP that involves support to computers to identify the sentiment behind written content or analyzed audio. Adding this ability to automatically detect sentiment in large volumes of text and speech opens up new possibilities for algorithm development (Ghosh and Gunning, 2019).

With the growth of Web 2.0 platforms such as blogs, discussion forums, social networks and various other types of media, consumers have within their reach the power to share their experiences and opinions, positive or negative, regarding any product or service (Pang et al., 2008).

In the current literature it is possible to find a variety of techniques used for sentiment analysis, such as machine learning and lexicon-based approach (Medhat et al., 2014).

Although these techniques are widely used, such as supervised and unsupervised learning, which use algorithms such as Naive Bayes (Rish et al., 2001), Bayesian Network (Kitson et al., 2021), Maximum Entropy (Wu, 2012) and linear classifiers with the use of Neural Networks (Abdi et al., 1999) and Support Vector Machine (Mammone et al., 2009), these approaches suffer from the problem of correct polarity classification. This means that mistakes made in the initial stages throughout the execution, influence the final quality of the result.

Obtaining better results in polarity classification is considerably important, considering the range of activities that make use of sentiment analysis results. Polarity concerns the classification of a text if it is a positive, negative or neutral citation. These activities carried out by companies and organizations bring important returns to society, whether in the form of

[a] https://orcid.org/0000-0003-2135-9947
[b] https://orcid.org/0000-0003-4176-566X
[c] https://orcid.org/0000-0003-2384-011X

Lopes, A., Gomes, V. and Zafalon, G.
Optimizing Natural Language Processing Applications for Sentiment Analysis.
DOI: 10.5220/0012632000003690

In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 1*, pages 698-705
ISBN: 978-989-758-692-7; ISSN: 2184-4992

monitoring the reputation of companies, quality of products and services, monitoring security and fraud (Pang et al., 2002).

Thus, the aim of this work is to obtain better results in Natural Language Processing with Sentiment Analysis in social networks, through the combination of different techniques, carried out in the following steps:

1. Linguistic pre-processing application for noise reduction, boosting the NLP processing action in order to facilitate the classification process.
2. Use of Machine Learning techniques with supervised learning and its main algorithms for data classification and sentiment analysis.

This work is organized as follows: in section 2, the related works are presented; in section 3, is described the development of the approach; in section 4, the obtained results are presented and analyzed; finally, in section 5, the conclusions are showed.

2 RELATED WORKS

In the literature, we find several works, both in the context of sentiment analysis using natural language processing, and in sentiment analysis using machine learning and deep learning.

2.1 Sentiment Analysis Using NLP

In (Qiu et al., 2010), a dictionary-based approach is used to identify the sentiment of sentences in the context of advertising. At work, an advertising strategy was proposed to improve the relevance of ads and the user experience. The authors worked with data from web forums. The results showed that the proposed model performed well in advertising keyword extraction and ad selection.

In (Hatzivassiloglou and McKeown, 1997), an initial list of opinion adjectives was used along with a set of linguistic restrictions to identify more opinion words and their orientations.

2.2 Sentiment Analysis Using Machine Learning

In (Kang et al., 2012), an optimized Naive Bayes classifier is used to solve the trend problem of greater accuracy in classifying positive samples (about 10% higher than in negative samples). This creates the problem of decreasing mean accuracy when the accuracy for the two classes is counted together. The work showed that using this algorithm together with a database of restaurant reviews it was possible to reduce the difference in accuracy between classes when compared to the traditional Naive Bayes and the Support Vector Machine (SVM). Recall and precision measures also improved.

In (Chen and Tseng, 2011), two SVM-based multiclass algorithms are used: One-against-All and Single-Machine Multi SVM to categorize comments. They proposed a method to assess the quality of information in analyzed products considering it as a classification problem. They also used an information quality framework to find the set of information-oriented attributes. They worked on reviews of digital cameras and MP3 players. The results showed that the method can correctly classify reviews in terms of its quality and that it significantly outperforms advanced methods.

The unsupervised approach was also used in (Xianghua et al., 2013) to automatically detect the aspects discussed in Chinese social networks and also the feelings expressed in different aspects. An LDA (Latent Dirichlet Allocation) model was used to uncover multi-aspect global themes from social commentary, then researchers extracted local theme and associated sentiment based on a scrolling window context over the text. They worked on social comments that were extracted from a blog dataset (2000-SINA) and a lexicon (300-SINA HowNet).

They also showed that their approach achieved good results in separating themes and improved the accuracy of sentiment analysis. The model also helped uncover several aspects of the topics and associated sentiment. (Ko and Seo, 2000) proposed a method that divides documents into sentences and classifies each sentence using the keyword lists of each category and a measure of sentence similarity.

In (Zharmagambetov and Pak, 2015), sets of decision trees are used to perform sentiment analysis on movie reviews. However, for extracting attributes from the text, they used deep learning methods. The methodology chosen was Word2Vec, which allows the capture of semantic characteristics of words. The vectors obtained by the feature extraction process were clustered by k-means. Each cluster, out of a total of 2000, had an average of 5 words. These were selected by their proximity in vector space. Clusters were used as inputs to the classifier. In the work, a reference database with movie reviews was used. The result of the experiment showed that the approach using feature extraction by deep learning was significantly better than the one using bag-of-words with 5000 entries.

2.3 Sentiment Analysis Using Deep Learning

Finally, deep neural networks were used in (Hu et al., 2015) for the analysis of sentiment in reviews of electronic products, movies and hotels. The authors created a classification framework that uses 3 different methods for attribute extraction: frequency-based, context-based, and part-of-speech tagging. Each method feeds a neural subnet that reduces the dimensionality of the attribute space. The outputs of these subnetworks feed the main network that is responsible for the analysis of feeling.

3 DEVELOPMENT

3.1 Data Collect

The collected data comprises a collection of tweets, each labeled with the sentiment expressed in the tweet, which can be positive, negative, or neutral. This dataset contains a diverse range of tweets, capturing opinions, emotions, and attitudes of Twitter users regarding various topics such as movies, products, events, or general experiences.

The utilized dataset consists of exactly 5,000 positive tweets and 5,000 negative tweets. The exact balance between these classes is not a coincidence; the intention is to maintain a balanced dataset.

3.2 Data Preparation

Data preprocessing aims to enhance the quality and effectiveness of the analyses and models that will be applied to the data.

Concatenating lists of data in Natural Language Processing (NLP) is a recommended technique in certain scenarios as it can improve data representation and provide additional insights for analysis or modeling. This technique is particularly useful when text data comes from various sources or different contexts and one wants to create a single, more diverse and comprehensive dataset.

In machine learning, including applications of Natural Language Processing (NLP) and Artificial Intelligence (AI), splitting the dataset into training and testing sets is a common and crucial practice.

In this specific case, we have chosen a 20% portion for testing and 80% for training, considering that the test set is relatively large, which can be beneficial when dealing with a sufficiently large dataset.

3.3 Data Preprocessing

Data preprocessing is one of the critical steps in any machine learning project. It involves cleaning and formatting the data before feeding it into a machine learning algorithm.

In this project, we utilized the following tasks:

- Tokenization. Tokenization involves dividing strings into individual words without white spaces or tabs.
- Lowercasing. In this step, we also converted each word in the string to lowercase.
- Removal of Stopwords and Punctuation. As we are working with Twitter data, we removed some commonly used substrings on the platform, such as hashtags, retweet tags, and hyperlinks. For this, we utilized the ”re” library to perform regular expression operations on our tweets. We defined a search pattern using the sub() method to remove matches and replace them with an empty character.
- Stemming. This process involves converting a word to its most general form or root. It helps to reduce the size of our vocabulary. For example, words like ”learn,” ”learning,” and ”learned” all derive from their common root ”learn.” However, in some cases, the stemming process produces words that are not correct spellings of the root word.

Since we are using the NLTK library, we have access to various modules for stemming. In this kernel, we used the Porter Stemmer. The objective of using this algorithm is to simplify words to their root form heuristically, meaning by following specific rules without relying on a complete dictionary.

Mapping each word-sentiment pair and its frequency is a crucial step in text processing for sentiment analysis tasks. This approach is commonly employed to represent text in a format suitable for applying machine learning algorithms or other text analysis techniques.

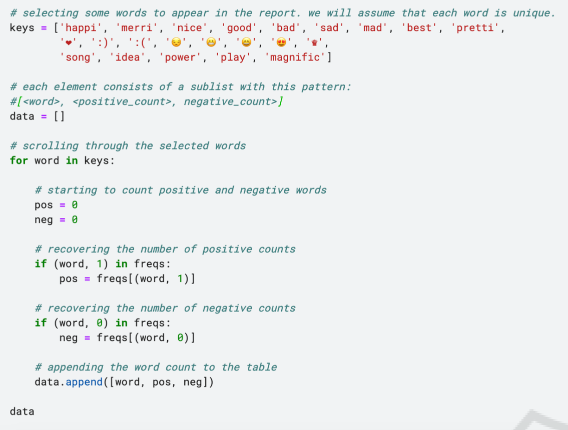
Moreover, it is necessary to extract features from the preprocessed data, such as word frequency and n-grams. These features will be used to train and test the machine learning models.

The Word Count Table, also known as a word histogram, is a tabular representation that displays the frequency of word occurrences in a text or set of texts.

Through this table, we will conduct a Descriptive Analysis, obtaining a descriptive view of the distribution of words found in the tweets.

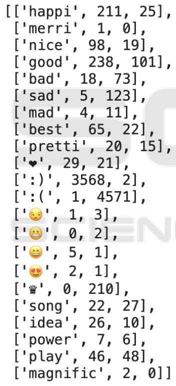
Thus, it is necessary to select a specific set of words for visualization. This process is recommended for text analysis and natural language processing situations and is often referred to as ”Keyword Analysis.” It can provide valuable insights into the content, topics, sentiments, or specific characteristics of the text.

Figure 1 illustrates the Python commands performed for Word Selection for Visualization.



```
# selecting some words to appear in the report. we will assume that each word is unique.
keys = ['happi', 'merri', 'nice', 'good', 'bad', 'sad', 'mad', 'best', 'pretti',
        '❤', ':)', ':(', '😒', '😬', '😄', '😍', '♛',
        'song', 'idea', 'power', 'play', 'magnific']

# each element consists of a sublist with this pattern:
#[<word>, <positive_count>, negative_count>]
data = []

# scrolling through the selected words
for word in keys:

    # starting to count positive and negative words
    pos = 0
    neg = 0

    # recovering the number of positive counts
    if (word, 1) in freqs:
        pos = freqs[(word, 1)]

    # recovering the number of negative counts
    if (word, 0) in freqs:
        neg = freqs[(word, 0)]

    # appending the word count to the table
    data.append([word, pos, neg])

data
```

Figure 1: Visualizing the words of Python commands.



```
[['happi', 211, 25],
 ['merri', 1, 0],
 ['nice', 98, 19],
 ['good', 238, 101],
 ['bad', 18, 73],
 ['sad', 5, 123],
 ['mad', 4, 11],
 ['best', 65, 22],
 ['pretti', 20, 15],
 ['❤', 29, 21],
 [':)', 3568, 2],
 [':(', 1, 4571],
 ['😒', 1, 3],
 ['😬', 0, 2],
 ['😄', 5, 1],
 ['😍', 2, 1],
 ['♛', 0, 210],
 ['song', 22, 27],
 ['idea', 26, 10],
 ['power', 7, 6],
 ['play', 46, 48],
 ['magnific', 2, 0]]
```

Figure 2: Python Commands for Word Selection for Visualization.

As described, there is a focus on relevant information. Some words such as ”good,” ”song,” and ”play” appear in both positive and negative word counts, indicating a dual sense of the word. Other words like ”happy,” ”nice,” and ”sad” stand out in their respective polarities, and the substantial count of punctuations and emojis, which depending on context, can be positive, negative, or neutral.

To provide a better illustration, let’s use a scatter plot to visually inspect this table.

We have 3568 counts in the positive area, while only 2 in the negative area. The red line marks the boundary between positive and negative regions. Words close to the red line might be classified as neutral. Figure 3 illustrates the Visualization of the Scatter Plot.

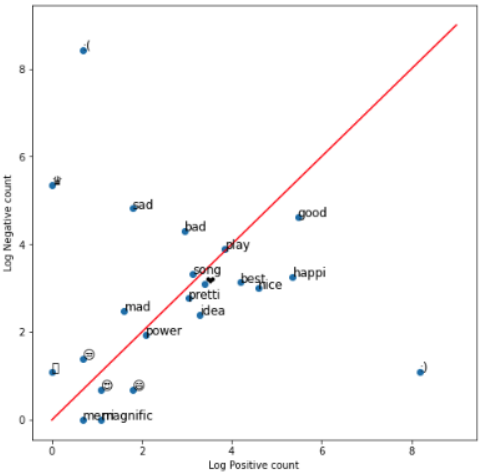


Figure 3: Scatter Plot.

According to (Bisong and Bisong, 2019), a frequency dictionary is a crucial tool in sentiment analysis, especially when using logistic regression as a classification method. It’s used to capture information about the frequency of words in relation to specific sentiments (positive, negative, neutral) present in the training data.

Now we need to process the tweets, tokenizing them into individual words, removing stopwords, and applying stemming. For this purpose, we use the function *process_tweet( )*.

3.4 Definition of Algorithm and Model

Defining an algorithm and model is a fundamental step to achieve accurate and reliable results. In this article, we have chosen the Logistic Regression algorithm to train and evaluate the results.

The choice of Logistic Regression was due to its ease of interpretability, which is particularly useful for understanding the importance of each variable. Additionally, its simplicity makes it quick to train and comprehend.

The model definition is represented by equations 1 and 2:

$$h(z) = \frac{1}{1+\exp^{-z}} \tag{1}$$

$$z = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \theta_N x_N \tag{2}$$

Within the model, it is necessary to define the Cost and Gradient Function (equation 3), which is the average of the logarithmic loss across all training examples. This is considered a fundamental step in optimizing logistic regression models, as well as in many other machine learning algorithms. It plays a crucial role in the model training step, where the objective is to adjust the model's parameters to minimize the error and enhance prediction performance.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h(z(\theta)^{(i)})) + (1-y^{(i)})\log(1-h(z(\theta)^{(i)})) \tag{3}$$

Where: - m is the number of training examples.

- $y^{(i)}$ is the actual label of the i-th training example.

- $h(z(\theta)^{(i)})$ is the model's prediction for the i-th training example.

In addition to the Cost and Gradient Function, it is necessary to define the loss function for a single training example. The goal is to measure the discrepancy between the predictions made by the model and the actual (label) values of the training data. This can be verified by equation 4:

$$Loss = -1 \times \Big(y^{(i)} \log(h(z(\theta)^{(i)})) + (1-y^{(i)})\log(1-h(z(\theta)^{(i)}))\Big) \tag{4}$$

Note that when the model predicts 1 ($h(z(\theta)) = 1$) and the label y is also 1, the loss for that training example is 0. Similarly, when the model predicts 0 ($h(z(\theta)) = 0$) and the actual label is also 0, the loss for that training example will be 0.

Another important step is updating the weights during the training of machine learning models. To update the weight vector θ, we apply gradient descent to iteratively improve the model's predictions.

The gradient of the cost function J with respect to one of the weights θ_j is, which can be verified by equation 5:

$$\nabla_{\theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h^{(i)} - y^{(i)})x_j \tag{5}$$

- i is the index over all m training examples.

- j is the index of weight θ_j, where x_j is the feature associated with weight θ_j.

To update the weight θ_j, we adjust it by subtracting a fraction of the gradient determined by α, using equation 6:

$$\theta_j = \theta_j - \alpha \times \nabla_{\theta_j} J(\theta) \tag{6}$$

The learning rate α is a value we choose to control the size of a single update.

3.5 Feature Extraction

Feature extraction involves identifying and selecting the most relevant and informative features or attributes from the original data to be used as inputs in the model.

We have a list of tweets that we need to extract and store in a matrix. The first extracted feature is the number of positive words in a tweet, and the second feature is the number of negative words in a tweet.

3.6 Model Training

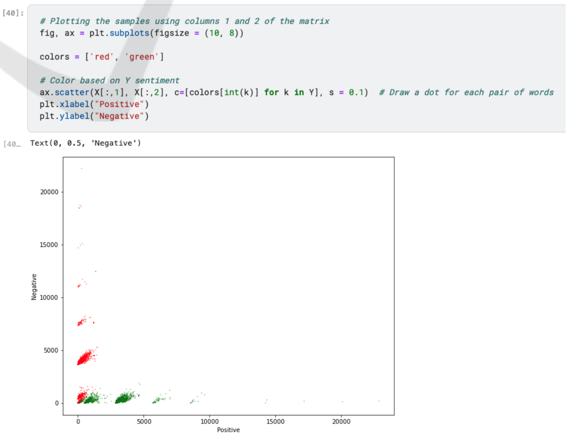
Model training is essential in machine learning and sentiment analysis. It teaches the model to recognize patterns in data, adjust parameters for accurate predictions, and generalize to new examples. This leads to better automated decisions, insights into data relationships, and performance optimization. The iterative training process adapts the model to the specific problem and allows continuous updates to maintain its relevance over time.

After feature extraction, we initiate the training process by stacking the features for all training examples into a matrix X.

3.7 Results Visualization

According to (Bisong and Bisong, 2019), visualization can aid in the selection of relevant features. By plotting scatter plots, correlation matrices, or bar charts, you can identify which features have the most influence on the outcomes and decide which ones to include in the model.

After stacking, the next step is to visualize the tweets and see how they are distributed along the X and Y axes. Figure 4 presents the Python commands for visualizing the samples.



```
[40]:
# Plotting the samples using columns 1 and 2 of the matrix
fig, ax = plt.subplots(figsize = (10, 8))

colors = ['red', 'green']

# Color based on Y sentiment
ax.scatter(X[:,1], X[:,2], c=[colors[int(k)] for k in Y], s = 0.1)  # Draw a dot for each pair of words
plt.xlabel("Positive")
plt.ylabel("Negative")
```

```
[40… Text(0, 0.5, 'Negative')
```

Figure 4: Python commands for visualizing the samples.

3.8 Model Evaluation

According to (Bisong and Bisong, 2019), evaluating the model is crucial to measure its ability to make accurate predictions on unseen data. This verifies whether the model generalizes well, avoiding overfitting. Evaluation reveals the actual performance of the model, aiding in adjusting hyperparameters and choosing the best path. It also helps identify issues like bias or systematic errors, providing insights for improvements and validating the practical utility of the model.

4 EVALUATION AND RESULTS

In this section, the testing methodology employed and the results obtained from the execution of the proposed method in this study are discussed. The performance of the algorithms is compared using evaluation metrics.

The evaluation metrics utilized include:

- Accuracy. To measure the proportion of correct predictions in relation to the total number of predictions made by the algorithm.
- Precision. To assess the proportion of true positives in relation to the total positive predictions made by the algorithm.
- Recall. To measure the model's ability to identify all positive examples in a dataset.
- F1 Score. To measure the harmonic mean between precision and recall.

4.1 Testing Platform

The platform used was Google Colab, along with Jupyter Notebook, including Python version 3.7 and all necessary packages, with the main library being NLTK.

The choice of Google Colab is due to its cloud-based nature, minimizing the time spent on setting up development environments and acquiring NLP libraries, thus expediting the project's initiation.

Moreover, Colab allows for the utilization of computational resources from the Google Cloud Platform, including access to GPUs and TPUs, which can significantly accelerate the training of NLP models, especially deep learning models. Colab employs interactive Jupyter notebooks, enabling the writing and execution of code in blocks.

4.2 Logistic Regression vs SVM Approach

In the applied testing methodology, two distinct test cases were conducted, involving variations in the

Table 1: Comparison of Performance: Logistic Regression vs. SVM - 5,000 Tweets.

Metric	Logistic Regression	SVM
Accuracy	0.835	0.827
Precision	0.858	0.826
Recall	0.952	0.993
F1-Score	0.903	0.902

Table 2: Comparison of Performance: Logistic Regression vs. SVM - 10,000 Tweets.

Metric	Logistic Regression	SVM
Accuracy	0.773	0.767
Precision	0.795	0.811
Recall	0.740	0.700
F1-Score	0.766	0.751

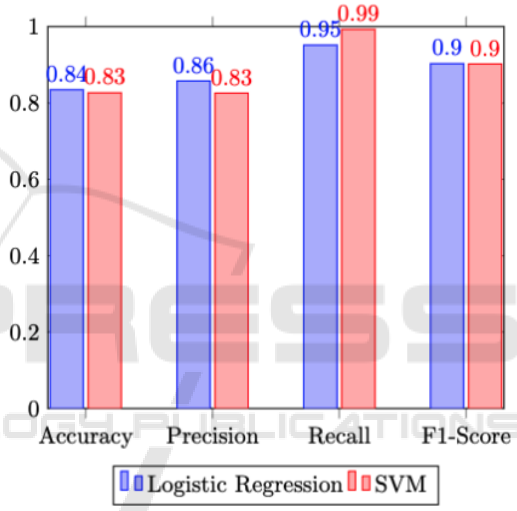


Figure 5: Performance Comparison (5.000 tweets).

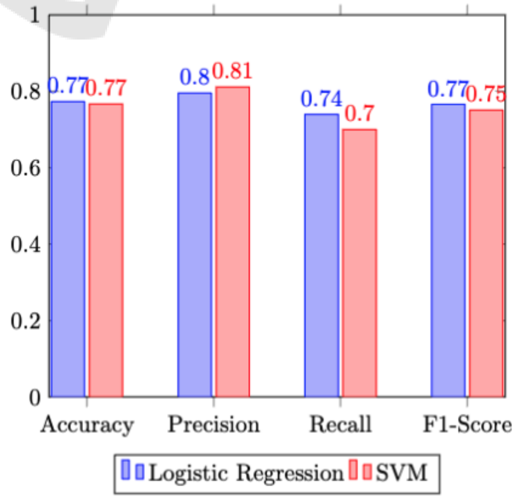


Figure 6: Performance Comparison (10.000 tweets).

number of words. Consequently, each test case underwent four iterations to derive metrics encompassing accuracy, precision, recall, and F1-Score. Initially, the input number of words was fixed at 5,000. Subsequently, this parameter was augmented to 10,000.

The outcomes of these method executions are presented in Tables 1 and 2.

Observations reveal that in the test case involving 5,000 words, the SVM approach produced suboptimal performance results. Nevertheless, upon implementing the strategy of increasing the word count to 10,000, only slight variations in the outcomes were noted.

Conversely, the Logistic Regression approach exhibited robust consistency, with results remaining largely unaffected despite the variance in word count.

Therefore, the significance of the Logistic Regression algorithm is underscored in methodologies where the dataset encompasses a larger number of words and demonstrates greater variation.

Therefore, Logistic Regression stands out as the number of words increases.

5 CONCLUSIONS

In conclusion, the importance of employing Supervised Machine Learning for Sentiment Analysis is underscored, providing a robust framework that delivers satisfactory results, particularly when handling extensive datasets.

Nevertheless, it is noteworthy that the field lacks a standardized computational method for Sentiment Analysis, resulting in outcome variations contingent on the specific techniques, algorithms, and models employed.

The utilization of algorithms such as Logistic Regression and SVM has proven instrumental in processing large volumes of textual data, markedly augmenting computational capabilities for such tasks. The outcomes of this study exhibit a level of performance comparable to published works in the field, signifying the efficacy of the proposed approach.

In essence, this method stands poised to aid professionals in the realm of computational intelligence engaged in Sentiment Analysis studies, offering a well-suited avenue for discerning polarities among analyzed words irrespective of the data source.

Finally, as future work, we will implement new machine learning method and compare them with the present ones. Moreover, we intend to establish comparisons with other approaches proposed in the literature, concerning the sentiment analysis issue.

ACKNOWLEDGEMENTS

The authors would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), under grant 88887.686064/2022-00, and São Paulo Research Foundation (FAPESP), under grant 2020/08615-8, for the partial financial support.

REFERENCES

Abdi, H., Valentin, D., and Edelman, B. (1999). *Neural networks*. Number 124. Sage.

Bisong, E. and Bisong, E. (2019). Logistic regression. *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pages 243–250.

Bonaccorso, G. (2017). *Machine learning algorithms*. Packt Publishing Ltd.

Chen, C. C. and Tseng, Y.-D. (2011). Quality evaluation of product reviews using an information quality framework. *Decision Support Systems*, 50(4):755–768.

Ghosh, S. and Gunning, D. (2019). *Natural Language Processing Fundamentals: Build intelligent applications that can interpret the human language to deliver impactful results*. Packt Publishing Ltd.

Hatzivassiloglou, V. and McKeown, K. (1997). Predicting the semantic orientation of adjectives. In *35th annual meeting of the association for computational linguistics and 8th conference of the european chapter of the association for computational linguistics*, pages 174–181.

Hu, Z., Hu, J., Ding, W., and Zheng, X. (2015). Review sentiment analysis based on deep learning. In *2015 IEEE 12th International Conference on e-Business Engineering*, pages 87–94. IEEE.

Hutchins, W. J. (2004). The georgetown-ibm experiment demonstrated in january 1954. In *Conference of the Association for Machine Translation in the Americas*, pages 102–114. Springer.

Kang, H., Yoo, S. J., and Han, D. (2012). Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications*, 39(5):6000–6010.

Khurana, D., Koli, A., Khatter, K., and Singh, S. (2022). Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, pages 1–32.

Kitson, N. K., Constantinou, A. C., Guo, Z., Liu, Y., and Chobtham, K. (2021). A survey of bayesian network structure learning. *arXiv preprint arXiv:2109.11415*.

Ko, Y. and Seo, J. (2000). Automatic text categorization by unsupervised learning. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

Mammone, A., Turchi, M., and Cristianini, N. (2009). Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):283–289.

Medhat, W., Hassan, A., and Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in information retrieval*, 2(1–2):1–135.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

Qiu, G., He, X., Zhang, F., Shi, Y., Bu, J., and Chen, C. (2010). Dasa: dissatisfaction-oriented advertising based on sentiment analysis. *Expert Systems with Applications*, 37(9):6182–6191.

Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Wu, N. (2012). *The maximum entropy method*, volume 32. Springer Science & Business Media.

Xianghua, F., Guo, L., Yanyan, G., and Zhiqiang, W. (2013). Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon. *Knowledge-Based Systems*, 37:186–195.

Zharmagambetov, A. S. and Pak, A. A. (2015). Sentiment analysis of a document using deep learning approach and decision trees. In *2015 Twelve international conference on electronics computer and computation (ICECCO)*, pages 1–4. IEEE.