Sentimental Analysis with Distributed Computation

Ching-Ting Tsai
Department of Mathematics
University of Waterloo
ON, Canada
c3tsai@uwaterloo.ca

Project Code

ABSTRACT

Distributed computation has become a widely applied solution across different domains as the volume of data is growing exponentially. This project focuses on the integration of distributed computing techniques in constructing a sentiment analysis model with millions of data sourced from Twitter. We developed our own algorithm for data preprocessing and text tokenization that are useful and meaningful to model training. The tokenized text data is further embedded into vectors of numerical values implemented with the Word2Vector model, serving as our training feature. We employ logistic regression, multilayer perceptron, and a simple neural network model for training and testing to compare and contrast their performance. However, further improvements are required to surpass the state-of-the-art models with the principle of distributed computing.

KEYWORDS

Distributed Computing, Big Data, Logistic Regression, Multilayer Perceptron, Neural Network

1 INTRODUCTION

In this technologically advanced era, the exponential growth of existing data largely involved sophisticated storage and computational solutions. The usage of data extends across diverse fields, including business analytics and the training of machine learning models. Traditional computation methods face constraints in storage space and computational time when processing extensive datasets. However, the emergence of data-intensive techniques, similar to the principles of divide and conquer, has demonstrated the ability to overcome these challenges.

This project focuses on leveraging the combination of Python, Jupyter, and Apache Spark to establish an interactive and visually engaging exploration system. With the utilization of machine learning models, including logistic regression, multilayer perceptron classification and neural network, our project attempts to conduct sentiment analysis on the Sentiment140 dataset, comprising 1.6 million tweets. The main objective is to explore the dynamic field of big data analytics, which involves integrating distributed computing with machine learning algorithms.

Apache Spark is a leading unified analytics engine that adopts the well-established MapReduce paradigm to distribute data across a cluster of machines. It also enables parallel processing and improves time and space efficiency in large-scale data operations. We heavily employed the Python API for Apache Spark (PySpark) to process and explore the extensive tweet dataset. This approach not only facilitates time and space-efficient data processing but also allows us to train the distributed data using various classification machine learning models, where logistic regression and multilayer perceptron are applied.

This project delves into the fundamental principles, challenges, and advancements in distributed computing, applied across various domains from academic research to industrial analysis. It also plays a vital role in the growing field of Artificial Intelligence, particularly in deep learning, which involves training large neural network models. As an outcome, we have developed a small interactive application that allows users to input sentences for sentiment analysis. It provides the output of the prediction, utilizing our trained models to determine whether the sentiment is positive or negative. This offers insight into the real-world application of distributed computing in sentiment analysis.

2 METHODOLOGY

This section presents the comprehensive methodology employed in the completion of the sentiment analysis model, integrating distributed computing with Apache Spark DataFrame and three different machine learning models. The application of these has significantly reduced the computing time and memory space, particularly when handling the extensive dataset of 1.6 million tweets.

2.1 Data Preprocessing

The initial and crucial step in data analysis and machine learning is data preprocessing. Given the size of the dataset, it's essential to dedicate effort to cleaning and preparing the data for later analysis. This process includes handling missing values, removing duplicated and unnecessary data, and standardizing the dataset. This ensures the quality and consistency of our data, laying the foundation for building a robust machine learning model and generating reliable results.

This project begins with a thorough exploration of the data using the Spark DataFrame. The schema is stored and printed to provide a detailed overview. Duplicate and null entries are systematically removed, and columns other than the label and text are excluded, as we focus on utilizing text as features and the label as the target variable for future model training.

We then develop algorithms for sentence cleanup, which involve converting all letters to lowercase, removing punctuation, and filtering out tags, hyperlinks, and letters appearing after apostrophes, as these elements contribute no meaningful information to the sentiment of the sentence.

After the cleanup phase, we proceed to tokenizing the cleaned text. This involves removing stop words using NLTK, filtering out words with fewer than 3 letters, and then stemming each word. This ensures words appear uniformly with the same root and enhances the consistency and meaningfulness of the dataset. The PySpark SQL API's UserDefinedFunction is leveraged throughout this process to efficiently manage distributed data processing within the DataFrame.

To gain insights in the structure of the dataset, we summarize the length of each word list, providing an overview of the upper and lower bounds of the data. Additionally, word frequency bars and word clouds are also generated for both positive and negative sentiments in order to give a deeper understanding of the most frequently occurring words in each category.

2.2 Word Embedding (Word2Vec)

To enable machine learning models in interpreting the meaning of sentences, it's crucial to transform text data into numerical vectors. We utilized the Word2Vec embedding architecture developed by Google to effectively vectorize our text. This technique allows us to represent tokenized words in a high-dimensional vector space which facilitates the calculation of semantic similarity and the relationship within sentences. The model is trained on a diverse word corpus, including Wikipedia, incorporating with two main algorithms: continuous bag-of-words (CBOW) and skip-gram [1]. These algorithms are widely recognized and commonly used in natural language processing applications.

2.3 Model Training

In addressing classification problems, many of the machine learning algorithms are available for consideration. Given our binary text classification task—categorizing between positive and negative sentiments—we focus on supervised models that are well-suited for binary classification. In this context, we implemented three prominent models in the field of machine learning: logistic regression, multilayer perceptron classification model, and a simple neural network model.

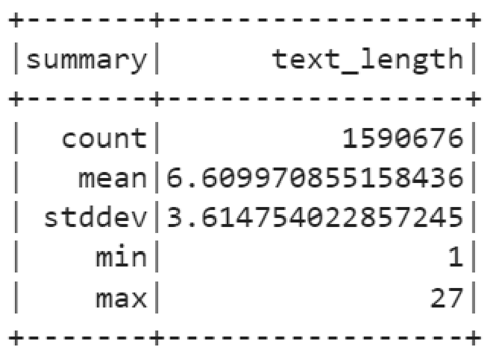
2.3.1 Logistic Regression. Research conducted by Prabhat and Khullar [2] indicates that logistic regression performs better than the Naïve Bayes classifier in terms of accuracy and precision for sentiment classification. As a machine learning algorithm, logistic regression applies the sigmoid function to produce binary outcomes based on given inputs and outputs. It is also recognized widely for its robust performance compared to other machine learning algorithms, which serves as a base-line supervised learning algorithm in the field of natural language processing [3][4]. Consequently, we decided to employ logistic regression as one of our sentiment analysis models for training our data.

2.3.2 Multilayer Perceptron Classification. This machine learning algorithm is often used in text classification, which adopts an artificial neural network architecture with layers of activation functions trained by back-propagation. In Spark's implementation, the algorithm utilizes the sigmoid function for hidden layers and employs the softmax function for the output layer to generate the predicted label [5]. It is implemented through Spark's ML library, a mdoern distributed library designed for distributed and parallel data processing [6].

2.3.3 Neural Network. In contrast to the distributed computation utilized by the previous two models with Spark's ML library, we constructed a simple feedforward network with one hidden layer employing ReLU activation and a final output layer using a Sigmoid activation for binary classification. As the model is built with PyTorch, it couldn't employ distributed computing while training and testing. And due to the limitation of memory space, we decrease our training and testing data size to 90% of the training data. The detailed performance results will be presented and discussed in the Results section.

3 DATA ANALYSIS

Data collection is a crucial starting point in model training, and the quality of the data significantly impacts the accuracy of the final analysis, potentially introducing bias into the results. Aligning with the goal of my sentiment analysis project, I chose to utilize the "Sentiment140 dataset with 1.6 million tweets" sourced from Kaggle as my dataset [7]. It is extracted through the Twitter API with essential feature fields including tweet polarity as the target variable and the tweet text as our feature data. Distributed computing appears to be an efficient approach for exploring and processing millions of data entries. Upon data exploration and cleaning, we discovered that the text length of each tweet tends to be relatively short, with an average length of approximately 7 words (Figure 1).



```
+-------+-----------------+
|summary|      text_length|
+-------+-----------------+
|  count|          1590676|
|   mean|6.609970855158436|
| stddev|3.614754022857245|
|    min|                1|
|    max|               27|
+-------+-----------------+
```

Figure 1: Summary of Text Length. This provides insight into the number of data entries and the minimum, maximum, and average text length after data cleaning.

In terms of data visualization, we have generated top 100 frequency words bar plots and word clouds for both positive and negative sentiment, as shown in the figures below. These visualizations provide insights in understanding the tone of the content, aligning with its target label of either positive or negative. They also allow us to identify key terms linked to each category, highlighting common details.

An interesting discovery arises as we observe that the top frequent words are commonly expressed in daily life, such as 'work,' 'sleep,' and 'love' (refer to Figures 2 and 3). However, this finding introduces a challenge when extrapolating our model to a more diverse testing dataset. For instance, if our model is applied to different genres of text, such as restaurant reviews or political views, the accuracy may be reflected as the usage and distribution of words in sentences are likely to differ.

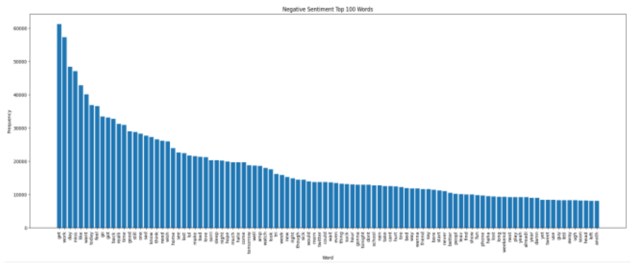


Figure 2: Words Frequency Bar (Negative). This bar graph illustrates the frequency distribution of the top 100 words within the negative sentiment text data. Each bar represents the occurrence of a specific word in the negative sentiment class.

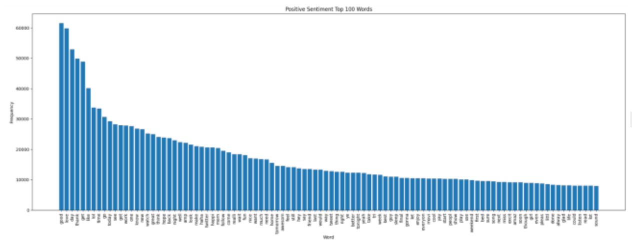


Figure 3: Words Frequency Bar (Positive). This bar graph illustrates the frequency distribution of the top 100 words within the positive sentiment text data. Each bar represents the occurrence of a specific word in the positive sentiment class.

Another challenge is associated with how language is used on social media platforms nowadays. Slang terms often exist in tweet text, which can potentially mislead the training model and cause it to misunderstand the meaning of sentences. For instance, in Figures 4 and 5 shown below, the slang term "lol" appears frequently. The high frequency of these informal expressions can affect the model's accuracy rate. Adding to the complexity, the use of sarcasm in tweet text can bring challenges for machine learning models to interpret correctly, while being clear for human beings. Misinterpretation becomes a factor that impacts the model's accuracy rates. Therefore, addressing these challenges can be difficult as they require complex solutions.

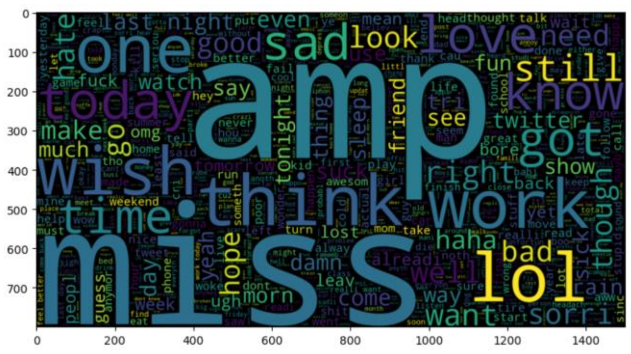


Figure 4: Word Cloud (Negative). This is a collection of words present in the negative sentiment text data, which shows the importance of each word within the negative sentiment class.

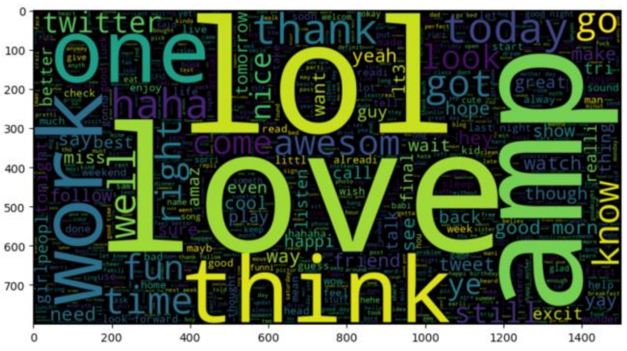


Figure 5: Word Cloud (Positive). This is a collection of words present in the positive sentiment text data, which shows the importance of each word within the positive sentiment class.

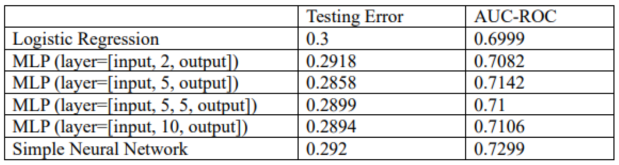
4 Sentiment Analysis Results

In the evaluation of our model, we calculated testing errors and the Area under the ROC Curve (AUC) to assess its accuracy and ability to distinguish between positive and negative sentiments. According to our results, as illustrated in Figure 6, the overall testing error ranges between 0.28 and 0.3, and the AUC lies between 0.6 and 0.73.

Upon conducting training and testing across all models, logistic regression appears to have the poorest performance with a testing error of 0.3. In contrast, multilayer perceptron presented better performance, especially with one hidden layer featuring five nodes, which achieves a testing error of 0.2858. Derived from this model, the confusion matrix is created to provide insight into how misclassified classes data are confused by the model [8]. In Figure 7, we observe that the True negative and False positive rate slightly differ by 0.2, which indicates the balanced of our model and demonstrates its proficiency in making fewer errors in both positive and negative predictions.

Our simple neural network, configured with one layer of 32 nodes with Relu activation, appears to perform slightly poorer testing accuracy than multilayer perceptron models; yet, it has the highest AUC result, showing its better capability in distinguishing between two sentiments. However, due to the limitation in heap memory, training and testing were conducted on only 90% of the numbers of training data.

Moreover, as the MLP involves layers of neurons in the training process, the capability of PySpark MLlib allowed it to train on the entire dataset, which signifies the power of the distributed computation. On the other hand, our simple neural network model faced the challenge of constraints in memory space, preventing it from being able to train and test on the complete dataset.



	Testing Error	AUC-ROC
Logistic Regression	0.3	0.6999
MLP (layer=[input, 2, output])	0.2918	0.7082
MLP (layer=[input, 5, output])	0.2858	0.7142
MLP (layer=[input, 5, 5, output])	0.2899	0.71
MLP (layer=[input, 10, output])	0.2894	0.7106
Simple Neural Network	0.292	0.7299

Figure 6: Sentiment Analysis Results. This table provides a comparative view of the testing error and AUC-ROC values calculated from machine learning models, including logistic regression, multilayer perceptron, and a simple customized neural network.

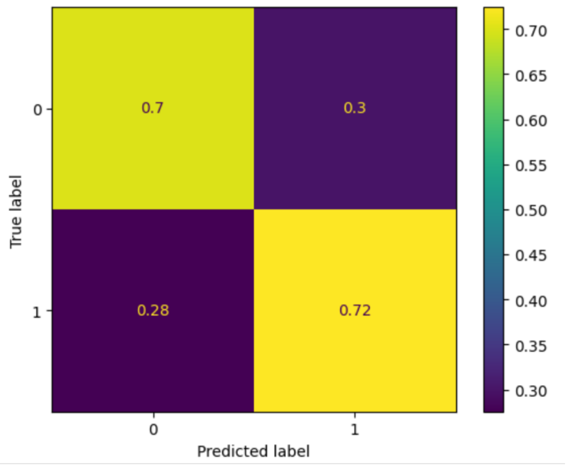


Figure 7: Confusion Matrix of Multilayer Perceptron Model. This confusion matrix presents the percentage of correct predictions for each true class on the multilayer perceptron model with one hidden layer of 5 nodes.

5 Discussion

This project has successfully integrated distributed computation with extensive sentiment text data for exploration and visualization using Apache technology. We cleaned up and prepared data with our tokenization algorithm, presented the top frequent words and word clouds for each positive and negative sentiment, and built machine learning models for sentiment analysis. However, our results are not performing as well as expected, with a testing accuracy of at least 0.75. Despite our low testing error in the result, there is still room for improving our models. The following are some potential ways for improvement:

1. Data Pre-processing:
   There are still opportunities to improve our project results at the phase of data pre-processing. One possible enhancement is to incorporate n-grams into our training features as it captures phrases in text that can be a challenge for models to comprehend the meaning. Another method that can be explored is to analyze the sentiment with the inclusion of

hyperlinks. In this project, we removed the link during the process of data cleaning; however, it could be beneficial in analyzing the sentiment if the links are retained and analyzed through web scrawling the page and including the text into our text vector for model training. Alternatively, we can analyze the sentiment of the link's content first and assign corresponding weights to refine our final model predictions.

2. Model Selection and Hyperparameter Tuning:
   Machine Learning models: Building an effective machine learning model requires extensive testing and hyperparameter tuning. If we explore more hyperparameter tuning strategies, we may lead to a greater result. Moreover, we can also advance our simple neural network by adding more hidden layers and nodes or incorporating advanced text classification techniques to improve our model's performance. For instance, we could implement transformer-based models such as Google's BERT, which has proved successful in the field of natural language processing and has already been pre-trained on a large corpus of text.

6 Conclusion

As data grows daily, the efficient processing of vast amounts of data for analysis or model training becomes crucial under the limitation in storage and time. The technology of distributed computing, which partitions data into clusters and process them concurrently across machines, is getting more popular and utilized widely in various fields. However, there is still much to discover in the application of big data processing, particularly in the field of AI, where massive datasets are essential for models training in reaching a higher accuracy.

In handling with millions of sentimental text dataset, the integration with the usage of the commonly used technology, Apache Spark, has significantly reduced the amount of time and space required for data exploring and processing, and model training. This approach allows us to present the visualization of useful information within the dataset and produce a high-performance text classification model. Although the result of our models failed to beat the state-of-the-art, there is still room to improve our overall performance through advanced feature extraction and model building techniques.

REFERENCES

[1] Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad. 2021. A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language Models. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 20, 5, Article 74 (September 2021), 35 pages. https://doi.org/10.1145/3434237

[2] Prabhat, Anjuman and Khullar, Vikas. 2017. Sentiment classification on big data using Naïve bayes and logistic regression, 1-5. 10.1109/ICCCI.2017.8117734.

[3] Cannannore Nidhi Kamath, Syed Saqib Bukhari, and Andreas Dengel. 2018. Comparative Study between Traditional Machine Learning and Deep Learning Approaches for Text Classification. In Proceedings of the ACM Symposium on Document Engineering 2018 (DocEng '18). Association for Computing Machinery, New York, NY, USA, Article 14, 1–11. https://doi.org/10.1145/3209280.3209526

[4] Anon. Speech and Language Processing (3rd ed. draft) Dan Jurafsky and James H. Martin. Retrieved December 8, 2023 from https://web.stanford.edu/~jurafsky/slp3/

[5] Anon. Classification and regression. Retrieved December 8, 2023 from https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier

[6] Piyush Sewal and Hari Singh. 2023. Analyzing distributed spark MLlib regression algorithms for accuracy, execution efficiency and scalability using best subset selection approach - multimedia tools and applications. (October 2023). https://doi.org/10.1007/s11042-023-17330-5

[7] KazAnova. 2017. Sentiment140 dataset with 1.6 million tweets. (September 2017). Retrieved December 8, 2023 from https://www.kaggle.com/datasets/kazanova/sentiment140/code?datasetId=2477

[8] Anon. Confusion matrix. Retrieved December 8, 2023a from https://www.sciencedirect.com/topics/engineering/confusion-matrix

[9] diniftikhar025. 2022. Sentimentanalysiswithlogisticregressioninpyspark. (August 2022). Retrieved December 8, 2023 from https://www.kaggle.com/code/diniftikhar025/sentimentanalysiswithlogisticregressioninpyspark