Analysis on NLP Models and their Applications

Kaiden Liu and Tao Shan
ky5liu@uwaterloo.ca, t4shan@uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

Abstract

Human language is crucial to the development of our current society. Researchers have been exploring different methods to interpret language, but we have yet to figure out a universal algorithm to perfectly understand human language (Chowdhary 2020). In this review paper, we compare the performance of XGBoost to other NLP models on sentiment analysis using the Sentiment140 dataset (Go, Bhayani, and Huang 2009), including steps in Explanatory Data Analytics and Natural Language Processing. We found that the BERT model achieved the highest accuracy of 84%, whereas the XGBoost model achieved an accuracy of 75% under a much shorter computation time. The result serves as a reference for any organizations that are interested in sentiment analysis to choose the most fitting model.

Introduction

There are an estimated 3.6 billion active users on social media in 2020 (Zachlod et al. 2022). The popularization of social media has generated a mass amount of text data for people to gain insights. Among the different types of research, sentiment analysis is one of the more popular methods to be used on social media data. With sentiment analysis, we could better understand how certain topics are perceived in society by classifying the emotion of the text data on social media. One of the biggest challenges of analyzing text data from social media (eg: Tweets from Twitter) is that they are full of errors with non-standard terms, which cause the data to be noisy and sparse (Derczynski et al. 2013). In this research paper, we explore the possibilities of applying XGBoost to NLP tasks. XGBoost is a scalable end-to-end tree boosting system and has shown great results in machine learning competitions (Chen and Guestrin 2016). The model's outstanding performance is due to its ability to handle sparse data (Chen and Guestrin 2016). On the other hand, Transformer based model is the current state-of-the-art model since the published of the paper "Attention Is All You Need" (Vaswani et al. 2017), and outperforms RNNs in many NLP tasks (Gillioz et al. 2020).

Many businesses could take advantage of NLP to assist them in making decisions. For example, finance and trading companies utilize NLP to understand the social consensus on a topic and adjust their strategies accordingly (Bahja 2020). Similarly, There have been great successes in the healthcare field, where they apply NLP models on health reports to search for indicators of diseases at early stages, such as PTSD and depression (Bahja 2020). Clearly, NLP models can be applied to have many usages, and identifying the most efficient model would be beneficial. Hence, our group compares the different models to see which performs best for sentiment analysis problems. There was a success in gaining insights from Twitter data sentiment analysis (Aramaki, Maskawa, and Morita 2011). In this research by Aramaki et al., they have successfully derived real-world insights from tweets using support vector machines. Aramki proposed a model to analyze tweets related to the flu to predict the incoming flu season (Aramaki, Maskawa, and Morita 2011). (Go, Bhayani, and Huang 2009).

There have been promising results with XGBoost on a machine learning competition website, Kaggle, where 17 winning solutions out of 29 solutions were achieved by XGBoost models in 2015 (Chen and Guestrin 2016). We believe that XGBoost could be a cheaper alternative for NLP tasks when we want to keep the cost low while achieving high accuracy. In order to better evaluate the performance of XGBoost, we compare the metrics of XGBoost with BERT. BERT, also known as the Bidirectional Encoder Representations from Transformers, developed by Google, is the current state-of-the-art model for NLP tasks (Xu et al. 2019). We are interested in whether XGBoost could produce a comparable result to BERT based on the model performance metrics and time efficiency. We picked the sentiment140 dataset as the out dataset to evaluate due to its large amount of data points (Go, Bhayani, and Huang 2009). The Sentiment140 dataset collected 1.6 million tweets; every tweet has a positive or negative label to indicate the tweet's emotion. We will further analyze previous standard models as the baselines, including Naive Bayes, Random Forest, Support vector machine, and K-nearest neighbours algorithm (Medhat, Hassan, and Korashy 2014). We use accuracy and F1 score as metrics to compare models' performance and training/testing time for time efficiency.

We assessed our result by splitting the training/testing data with 80%/20% percentages.

Main Results Our research result shows BERT shows much better performance than any other models that our group compared. XGBoost has a similar performance to SVM and is better than other models, including KNN, Naive Bayes and Random Forest.

Contributions This paper is the first paper to apply XGBoost and BERT in Twitter sentiment analysis data.

- We collected the dataset on the Kaggle Platform, which the dataset is generated by Twitter API.
- We implemented the algorithms for XGboost and BERT models.
- We compared the performance of XGBoost and the BERT algorithm with other baseline models on Twitter sentiment analysis data and achieved the best performance by the BERT algorithm.
- Our implementation for the Hyper-pruning parameter improves accuracy for both XGBoost and BERT by at least 3%, compared with the initial model setting.

Related Work

Previous work of NLP focused on understanding emoticons, hashtags, and acronyms within tweets using parts-of-speech (POS) features, WordNet, and tree representations.(Agarwal et al. 2011) The result indicates that Unigram + Semi-features has the best performance of 75.39% accuracy. Another article discussed various feature types for sentiment extraction were evaluated, including punctuation, patterns, words, and N-grams. They concluded that Twitter Hashtags and smileys perform the best when trying to distinguish between sentiment types (Davidov, Tsur, and Rappoport 2010)

Huq et al. conducted a sentimental analysis using KNN (k-nearest neighbour) and SVM (support vector machine) with different feature choices such as n-grams, word features, and pattern features (Huq, Ali, and Rahman 2017). Huq et al. concluded that KNN performs better than SVM with an accuracy of 84%. Similarly, Dey et al. discussed the performance of the two algorithms Naive Bayes and KNN to perform sentiment analysis on the IMDB dataset (Dey et al. 2016). Naive Bayes achieved a far better result than KNN on overall accuracy, precision, and recall metrics. From our observation, before the rise of transformer models, KNN, Naive Bayes, Random Forest, and SVM were very popular among NLP researchers. Hence, we also include these models as further references.

XGBoost is a scalable end-to-end tree-boosting system widely used on the machine learning competition website, Kaggle. We believe that XGBoost could perform well with NLP data due to its ability to treat sparse datasets effectively. Afifah et al. applied XGBoost on a telemedicine app reviews dataset and achieved an impressive result of 96.42% accuracy (Afifah, Yulita, and Sarathan 2021). The sentiment analysis was conducted to examine customers' opinions on using a telemedicine application to see how they could improve their app design (Afifah, Yulita, and Sarathan 2021). The XGBoost includes hyper-parameter pruning for learning rate, number of trees built in the model, and maximum tree depth. In the end, Afifah et al. concluded that (Afifah, Yulita, and Sarathan 2021)

Bert algorithm (Bidirectional Encoder Representations from Transformers) performs well on a wide range of tasks at both the sentence and token levels (Devlin et al. 2018). Xu et al. proposed a new task that adopted BERT as their base model to review reading comprehension and aspect-based sentiment analysis (Xu et al. 2019). The paper demonstrated BERT's ability to understand text data by applying many different reviews dataset. The BERT-inspired method achieves great accuracy after fine-tuning the parameters (Xu et al. 2019).

Methodology

A. describe the algorithm

Related works have proven the effectiveness of both XGBoost and BERT on sentiment analysis tasks. In this research paper, we compare XGBoost and BERT on sentiment analysis on the Twitter dataset, sentiment140.

XGBoost is a scalable machine-learning system for tree boosting. Currently, the system is available for open-source package (Chen and Guestrin). Among 29 winning solutions published on Kaggle's blog during 2015, 17 solutions used XGBoost (Chen and Guestrin). For comparison, the second most popular method, deep neural nets, was used in 11 solutions. XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function and a penalty term for model complexity. The training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. XGBoost has a wide range of applications, including classification, regression problems, ranking, and user-defined prediction problems. XGboost has the advantage that it needs fewer machines to run the entire dataset (Chen and Guestrin), so it's effective for large datasets. It also works well when data is nonlinear, non-monotonic, or with segregated clusters.

BERT is a model for Bidirectional Encoder Representations from Transformers, which is designed to pre-train deep bidirectional representations from the unlabeled text by jointly conditioning on both left and right context in all layers(Devlin et al. 2019). BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. The paper's results show that a language model which is bi-directionally trained can have a deeper sense of language context and flow than single-direction language models (Devlin et al. 2019). BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in (Vaswani et al. 2017) and released in the tensor2tensor library.

B. describe how each algorithm works in detail
B.1 XGBoost

The idea for XGBoost is an Extreme gradient boost, which combines many weak learners to strong learners. First, the prediction score is summed up by each tree's prediction score. Let f_k be the k-th tree's prediction function,

Start with training the first tree, $\hat{y}_i^{(0)} = 0$
Then, because $\sum\limits_{k=1}^{t-1} f_k(x_i) + f_t(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$ when adding t-th tree into the model,
$\hat{y}_i = \sum\limits_{k}^{K} f_k(x_i)$ (Li, Li, and Gao 2020)

XGBoost, which uses additive training methods, fixes one tree each time and learns one new tree at a time. Each time for training the model, we need to find the leaf to optimize our objective function.

$$obj = \sum_{i=1}^{N} l\big(y_i, \hat{y}_i\big) + \sum_{k=1}^{K} \Omega\big(f_k\big) \text{ (Li, Li, and Gao 2020)}$$

Ω, which is a penalty function, some examples such as l1 and l2 regulations. Then according to the proof by l2 regulations and using the Taylor expansion formula(Qiu 2019), the result for weight and the best objective reduction becomes
$weight_j^* = \frac{G_j}{H_j+\lambda}$
$obj^* = \sum\limits_{j=1}^{T} \left[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2\right] + \Upsilon T$

B.2 BERT

Here are the implementation steps for BERT:
1. pre-training BERT
The model predicts only those 20% tokens that were masked at random in the input tokens, in which case only the masked tokens are predicted by the model.

2. next sentence prediction
In order to select the sentences A and B for each pre-training example, it is important to consider the relationship between words and sentences, 50% of the time, B is the actual next sentence that follows A, and 50% of the time, it is a random sentence from the corpus. (Devlin et al. 2019)
Pre-training corpus trained 2500M words (Devlin et al. 2019), which is very powerful for understanding the word corpus.

BERT has a wide range of applications, such as language classification, question answering and sentiment analysis. BERT has a great model performance, the ability to process larger amounts of text and language and simple-to-use pre-training models.

C. Explain why to select these algorithms

In contrast to existing systems, XGboost scales far beyond billions of examples using far fewer resources than tree boosting. (Chen and Guestrin) With XGboost, unscaled data can be handled, and the performance and speed of the model can be improved. Specifically, in other research for sentiment analysis, the XGBoost algorithm is responsive to the features and has excellent accuracy and performance. (Nsaif and Abd 2022) So we think XGboost is an asset for us to discover the Twitter sentiment analysis dataset since no article has used XGboost in this dataset before.

BERT also has advantages for its accuracy and efficiency. BERT was trained on a large text corpus, which better understands text corpus and words. (Devlin et al. 2019)

Specifically, in our dataset, XGboost trains fast and has a high performance, with parallel training, regularization, and supports large datasets. For our group's computational power (which will mention in the result), XGBoost is the only model that can support training with the whole dataset. BERT is training slowly since it has too many neurons and hidden layers. But the efficiency of the model is far more than the normal machine learning models, and it's designed as a language model.

D. Parameter Values

For both the XGBoost and BERT models, we choose the following parameters at the beginning:

Parameters	Settings
tree_method	GPU_hist
max_depth	6
lambda	1
alpha	0

Table 1: Xgboost Parameter Initial Setting

The tree method constructs the algorithm used in the XGBoost, including the exact greedy algorithm, approximate greedy algorithm and faster histogram optimized approximate greedy algorithm. (Chen and Guestrin) GPU hist setting supports distributed training by GPU in a faster histogram-optimized approximate greedy algorithm, which fits the requirement of computational efficiency to deal with the large text analysis dataset. Max-depth controls over-fitting by setting a maximum depth of the tree growth. A large value may cause an increase in computational cost, and a low value causes imperfect information within the tree. So our group choose the value of 6, which controls over-fitting and contains most of the information.

L1 regularization: $Loss = Error(Y - \widehat{Y}) + \lambda \sum_1^n |w_i|$
L2 regularization: $Loss = Error(Y - \widehat{Y}) + \lambda \sum_1^n w_i^2$

Alpha is the L1 regulation, and lambda is the L2 regulation term on weights. The greater the value, the more regulation will apply to the model weight. The L-1 Regulation selects variables according to the penalty applied to the L1-norm. (Park and Hastie) The size of the L-2 regulation avoids the over-fitting issue.

For the number of estimators, learning rate and subsamples, our group will decide the value after pruning the parameters.

For BERT, our group chose the parameters suggested by the original academic paper(Devlin et al. 2018). We choose the parameter values for the number of epochs between 2 and 4 to satisfy the paper's suggestion. Also, our group will test the learning rate values during the fine-tuning.

E. Evaluate the Performance
Our group split the data for 80% training data and 20% testing data and evaluated the model in the testing dataset by Accuracy, F-1 Score, Log-loss, and Model execution time. Accuracy shows how accurately the model is trained on the system.

F. Dataset
Our group used the Twitter dataset includes 1, 600, 000 tweets were extracted with Twitter API, which split into 50% positive tweets and 50% negative tweets. We chose to use this dataset because it's generated by twitter's open API, with enough size for us to analyze. The dataset contains the user ID, the user's tweets and the label of tweets, either positive or negative. The tweets are sentences with all types of characters, such as punctuations and different tenses of words.

G. Data Preprocessing
For data preprocessing methods, we choose several methods of preprocessing text data, including lowercase, stop word removal, stemming, Dealing with Emojis and URLs, and removing numbers and punctuations. These preprocessing methods all remove noise or unhelpful parts of the data.(Pahwa, S., and Kasliwal 2018) Also, we encode the data by Tfidf-vectorizer for XGBoost and BERT encoder that fits with BERT algorithm

English stop words are chosen to remove by the NLTK package. NLTK is a Python library for analyzing human language data.

When dealing with URLs, first, our group detects the URL structures such as starting with "https", "www", and ending with 'com', 'edu' and other patterns in appendix 1. Then, our group deletes common structures from the URL. Such as converting 'www.overleaf.com' to 'overleaf' to maintain some useful information from the URLs.

To convert the emojis to the specific words that it applies. Such as, ":)" means smile. Other patterns are shown in appendix 2.

In order to evaluate the importance of a word in a file set, the TF-IDF (Term frequency-inverse document frequency) method is used. In a document, a word's importance is proportional to how often it appears and inversely proportional to how often it appears in the entire set of documents. (Qi 2020)

$TF_{i,j}$ means the frequency of words/corpus W_i in the paragraphs S_j. $n_{k,j}$ means number of words for words/corpus W_i in paragraphs S_j.

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

The IDF represents the entry with the inverse document frequency index. (Qi 2020) It equals the total number of paragraphs divided by the total number of paragraphs, including the word/corpus W_i

$$IDF_i = \log(\frac{|S|}{|S| s.t W_i \in S})$$

So, for each word/corpus W_i, the value assigned to the word/corpus is:

$$\text{TF-IDF}_i = TF_{i,j} \times IDF_i$$

So TF-IDF includes the information for both frequencies in each paragraph and the frequency among all paragraphs. In our model, we choose to include both uni-grams and bi-grams.

Results

In this section, our group presents the steps and results for model comparison of XGBoost, BERT, and other baseline models. We also show the dataset, design of the system, data exploration, preprocessing steps, detailed results, comparison with prior work and challenges.

A. Dataset
Our group used the Twitter dataset includes $1, 600, 000$ tweets extracted with Twitter API, which split into 50% positive tweets and 50% negative tweets.

B. Design and Goal of the system
Our group's system design shows in Figure 1, which includes detailed preprocessing steps, model choices, split train/test data, choosing model parameters, and Performance Evaluation. The reason for these steps is discussed in the methodology part.

The goal of the system design is to find the best model for sentiment analysis. Consumer needs and desires can be better understood using sentiment analysis as a new marketing technology.(Ciocodeică et al. 2020) In order to improve customer communication and develop more relevant messages, we want to ensure the prediction is accurate and time-efficient.

C. Data Exploration Results
Word Cloud is a powerful data visualization tool that summarizes words with high frequency. When a word appears bigger and bolder, it's more likely to be mentioned within a given text and is more important. In Figure 2, We

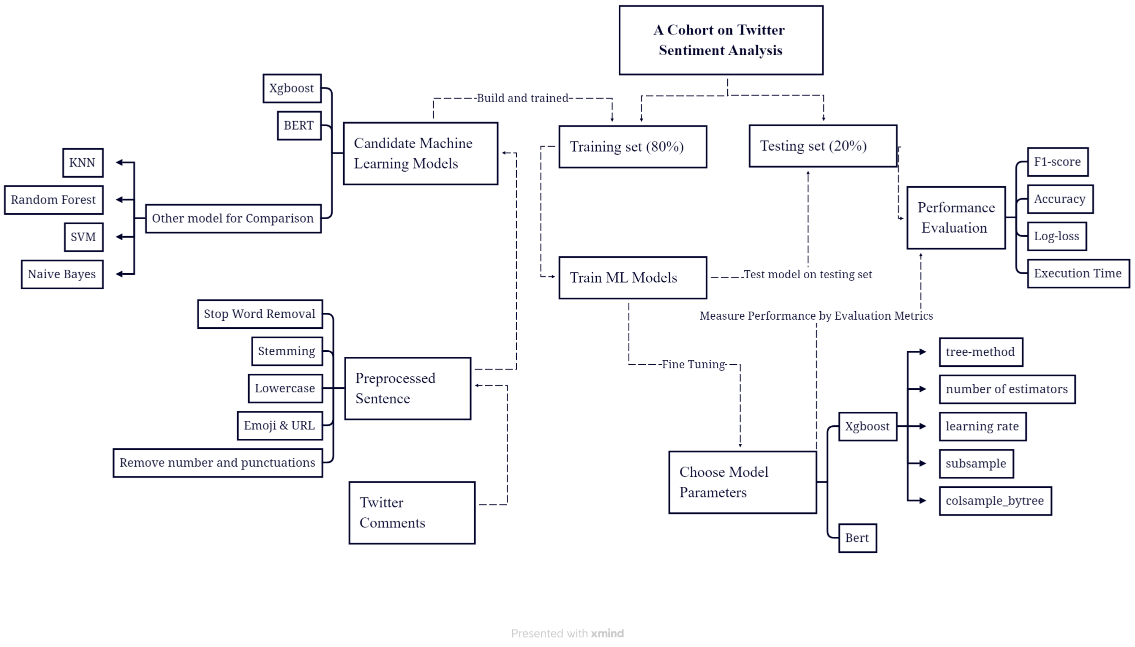

Figure 1: UML diagram for workflow

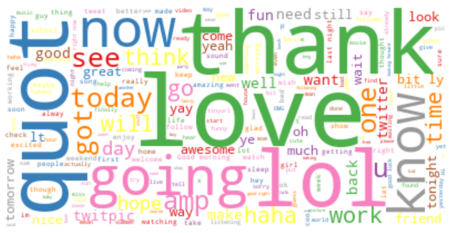

Figure 2: Word Cloud for Positive Words

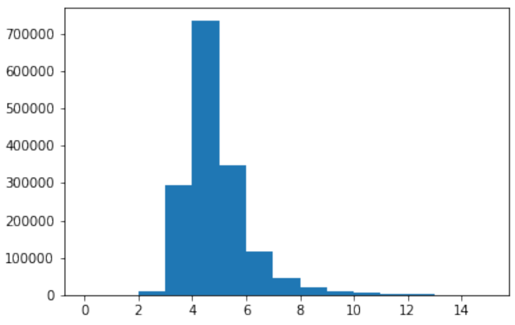

Figure 4: Histogram for sentence length

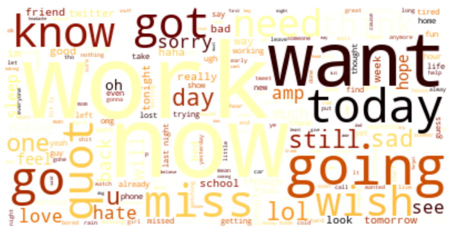

Figure 3: Word Cloud for Negative Words

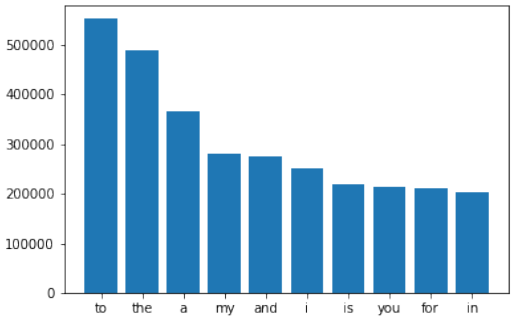

Figure 5: Bar Chart for top stopwords

found that words such as 'thank' and 'love' frequently show in positive words, and in Figure 3, words such as 'miss' and 'want' frequently show in negative words. The plots let us understand what kinds of words are within our data.

The histogram for sentence length (Figure 4) shows most tweets reply a word lengths between 3 and 6, which are all short sentences. We choose to use a word tokenizer in the preprocessing steps since the number of words is not too many, which keeps a small vocabulary size. When analyzing sentiment in small vocabulary sizes, the word tokenizer provided the best results. (Alyafeai et al. 2022)

The bar plot (Figure 5) shows that stop words have a very high frequency within our dataset. Reducing stop words in sentiment analysis helps with noise reduction and facilitates feature extraction. (Asghar et al. 2014)

D. Hyper-parameter analysis

In this article, our group focuses on XGBoost and BERT models. The hyper-parameter analysis is essential because the model needs to be tuned to fit a specific dataset. But the hyper-parameter process is extremely computationally expensive. (Yang and Shami 2020) Our group compares different essential parameters in these models and makes sure the parameter fits with the model well.

D.1 XGBoost In previous parts, our group discussed different parameters' usages. We compared both accuracy and log loss between the original settings and the new settings.

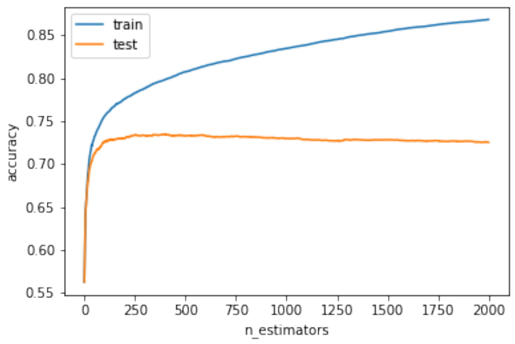

Figure 6: Xgboost: Accuracy and Number of estimators

According to Figures 6 and 7, choose the number of estimators = 388 to minimize both of accuracy and loss. Similarly, for other parameters, we tried the parameters in Table 2, including the number of estimators, learning rate and subsample. The final chosen parameter is shown in Table 3, including the number of estimators = 388, learning rate = 0.22, and subsample = 0.5.

D.2 BERT

Similar to BERT, we compared both accuracy and loss between the original settings and the new settings. Due to the limit of computational power, we tried the parameters that shown in table 4, including the learning rate and the number of epochs. The final chosen parameter is shown in Table 5, including the number of epochs = 2 and the learning rate = $5e^{-5}$.

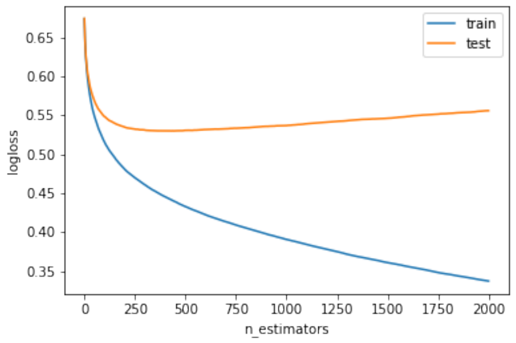

Figure 7: Xgboost: Log loss and Number of estimators

Parameters	Settings
Number of Estimators	[0, 1, 2, ..., 2000]
Learning Rate	[0.01, 0.02, ..., 0.30]
Subsample	[0.01, 0.05, 0.1, 0.15, ..., 0.5]

Table 2: Table for Xgboost Hyper-Parameter Pruning

Parameters	Settings
Number of Estimators	388
Learning Rate	0.22
Subsample	0.5

Table 3: Result for Xgboost Hyper-Parameter Pruning

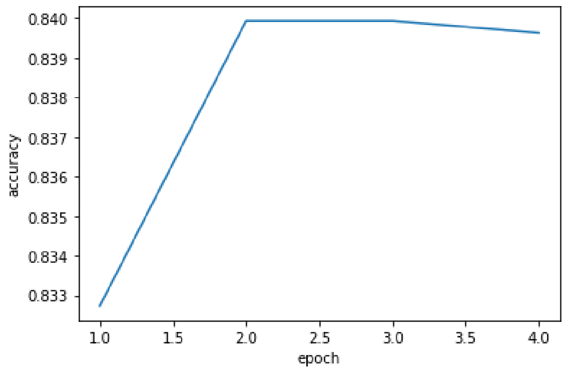

Figure 8: BERT: Accuracy and Number of epochs

E. Model Performance & challenges

In table 6, after fine-tuning the models, the comparison of model performance shows BERT's performance is better than other machine learning models, among F-1 score, accuracy and log loss. BERT achieved an F-1 score of

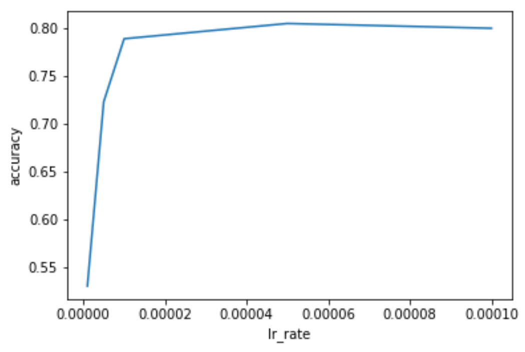

Figure 9: BERT: Accuracy and learning rate

Parameters	Settings
Number of Epochs	[1, 2, 3, 4]
Learning Rate	[$1e^{-6}$, $5e^{-5}$, $1e^{-5}$, $5e^{-4}$, $1e^{-4}$]

Table 4: Table for BERT Hyper-Parameter Pruning

Parameters	Settings
Number of Epochs	2
Learning Rate	$5e^{-5}$

Table 5: Result for BERT Hyper-Parameter Pruning

Techniques	F-1 Score	Accuracy	Log Loss
KNN	0.676	0.649	12.137
Naive Bayes	0.741	0.735	9.162
Random Forest	0.735	0.691	10.688
SVM	0.752	0.742	8.896
XGBoost	0.750	0.736	9.110
BERT	**0.840**	**0.840**	**0.424**

Table 6: Summarizing the model performance

Techniques	Model Fitting Time	Prediction Time
KNN	**0.009**	0.818
Naive Bayes	0.017	**0.008**
Random Forest	4.204	1.594
SVM	18.855	1.724
XGBoost	1.458	0.181
BERT	1195.300	44.700

Table 7: Summarizing the model efficiency

0.840, an accuracy of 0.840 and a log loss of 0.424, while XGboost achieved an F-1 score of 0.750, an accuracy of 0.736 and a log loss of 9.110. This means the BERT algorithm performs better than XGBoost for accurately predicting the result and less information loss.

In table 7, we found KNN, Naive Bayes and XGBoost have a low model fitting time and prediction time. But BERT's model fitting time and prediction time are far more slowly than other models since it has a more complex model structure. Compared with XGboost and BERT, XGBoost has a model fitting time of 1.458 seconds and a prediction time of 0.181 seconds. While BERT model trains slowly and requires a very large RAM to run this model, which needs 1195.300 seconds to fit the model and 44.700 seconds to predict the result.

The result is based on 100,000 data points, with GPU P100 and 83.48 GB RAM. At first, we have 13 MB RAM that supports 5,000 data points for BERT, then we buy the Google Colab Premium to support a larger dataset. So the challenge we met to produce the result with all data points is the BERT model needs a large RAM and a large computing unit to run, which cannot support all the data points to run.

F. Comparison with Prior Works

Compared with the model efficiency in the previous article (Kouloumpis, Wilson, and Moore 2021), which also did Twitter sentiment data analysis on a larger dataset than our sample, the best model's performance is similar to our XGBoost model (accuracy = 0.75), but our group's BERT model (accuracy = 0.84) is much better than this article by using fewer data.

G. The lesson we have learned

By implementing XGBoost and BERT algorithms in the sentiment analysis dataset, our group learned the more complex the model is, the more memory usage and higher CPU/GPU efficiency need to be used. When our group used different RAM quotas, CPUs and GPUs to train XGBoost, we found the efficiency was very different. We compare the results with the same system and the same data for all the models to keep the efficiency consistent.

Discussion

A. Interpret the result

Our research question was XGBoost, and the BERT model has a better performance than other models. The result shows BERT has an accuracy of 0.840, an F-1 score of 0.840 and a log loss of 0.424, which shows much better performance than any other models that our group compared. XGBoost has a similar performance to SVM and is better than other models, including KNN, Naive Bayes and Random Forest.

The BERT model's execution time is much longer than other models, which means if we have much more data to use, the BERT model cannot support to use of all the data. Though XGBoost has a worse result than BERT with our current subset of data, the model does not require a large RAM and runs fast. XGBoost has good time efficiency, with 1.458 seconds for model fitting and 0.181 seconds for model training.

B. Discuss the implication

Our groups used BERT and XGBoost models that were different from previous approaches. This article (A. and Priya 2020) discussed different approaches, such as Naive

Bayes and other deep learning approaches. In our study, compared with these approaches, BERT defeated all of these models based on our findings. XGBoost is similar to SVM for the model performance, but the time efficiency is far more efficient than SVM. Naive Bayes also has a low execution time, but the model performance is lower than other models.

C. Limitations

Due to the limited computational resource we had, we were unable to test all possible hyper-parameters. We choose some of the values in different parameters to get an optimal parameter for our data to use. If the computational power is enough, our group will test all possible parameter combinations to get our optimal model.

Another problem is that the BERT model needs a very long time and a large RAM to execute. Our dataset uses 1195 seconds for running one time and uses the maximum dataset that supports our system's memory. We must limit the data size to make the model available for our system to run and tune the parameters.

The last problem is with our system and data size. Our initial system only has 13 GB RAM that supports 5,000 data points for BERT and 100,000 data points for Naive Bayes, Random Forest and SVM. Only XGBoost and KNN support all the data with 1,600,000 data points. We choose to buy the Google Colab Premium and improve the maximum data point for the BERT algorithm to run from 5,000 to 100,000. Thus, we keep the data size of 100,000 for all experiments in this article. We spent all 100 compute units for the premium plan to train the possible parameters in BERT.

Overall, if our team has a better cloud system with a faster GPU and higher RAM, then more parameters will be tried, and a larger dataset will be used. So, better performance for all models will be shown. We can expect that the BERT algorithm will still have the best performance if there is no limit for the computing system.

Conclusion

We presented results for sentiment analysis on Twitter. We use previous standard models as our baseline and report all models' efficiency based on accuracy, F-1 score, Log loss and computational cost. Our work focuses on XGBoost and BERT's model performance, including data exploration, data preprocessing, model evaluation, and hyperparameter tuning. We conclude that the BERT model performs best in Twitter sentiment analysis, but its computational cost is high. XGBoost's model has a lower performance than BERT, but it still has a highly effective performance with a lower computational cost.

In future works, we will try more models, such as RNN and LSTM, as these are the previous state-of-the-art models. Due to the limitations of the computational cost of our machine, we focused more on BERT and XGBoost models in this article. We are also interested in applying our model to more data to extract values for businesses in different domains. In particular, we are interested in comparing SP500 and our sentiment analysis result to see if any relationship exists between social consensus and market trends and cycles. We believe that many valuable insights can be derived from social media, and we hope to use the stock analysis to support our idea further.

References

A., P., and Priya, K. S. 2020. Sentiment classification of tweets in twitter using cnn and dropouts in rnn. *International Journal of Computer Applications* 175(33):1–5.

Afifah, K.; Yulita, I. N.; and Sarathan, I. 2021. Sentiment analysis on telemedicine app reviews using xgboost classifier. In *2021 International Conference on Artificial Intelligence and Big Data Analytics*, 22–27.

Agarwal, A.; Xie, B.; Vovsha, I.; Rambow, O.; and Passonneau, R. 2011. Sentiment analysis of Twitter data. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, 30–38. Portland, Oregon: Association for Computational Linguistics.

Alyafeai, Z.; Al-shaibani, M. S.; Ghaleb, M.; and Ahmad, I. 2022. Evaluating various tokenizers for arabic text classification. *Neural Processing Letters*.

Aramaki, E.; Maskawa, S.; and Morita, M. 2011. Twitter catches the flu: detecting influenza epidemics using twitter. In *Proceedings of the 2011 Conference on empirical methods in natural language processing*, 1568–1576.

Asghar, D. M.; Khan, A.; Ahmad, S.; and Kundi, F. 2014. A review of feature extraction in sentiment analysis. *Journal of Basic and Applied Research International* 4:181–186.

Bahja, M. 2020. Natural language processing applications in business. *E-Business-Higher Education and Intelligence Applications*.

Chen, T., and Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Chowdhary, K. 2020. Natural language processing. *Fundamentals of artificial intelligence* 603–649.

Ciocodeică, D.-F.; Chivu, R.-G. P.; Popa, I.-C.; Mihălcescu, H.; Orzan, G.; and Băjan, A.-M. D. 2020. The degree of adoption of business intelligence in romanian companiesmdash;the case of sentiment analysis as a marketing analytical tool. *Fundamentals of artificial intelligence* 603–649.

Davidov, D.; Tsur, O.; and Rappoport, A. 2010. Enhanced sentiment learning using Twitter hashtags and smileys. In *Coling 2010: Posters*, 241–249. Beijing, China: Coling 2010 Organizing Committee.

Derczynski, L.; Ritter, A.; Clark, S.; and Bontcheva, K. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the international conference recent advances in natural language processing ranlp 2013*, 198–206.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Dey, L.; ; Chakraborty, S.; Biswas, A.; Bose, B.; and Tiwari, S. 2016. Sentiment analysis of review datasets using naïve bayes‘ and k-NN classifier. *International Journal of Information Engineering and Electronic Business* 8(4):54–62.

Gillioz, A.; Casas, J.; Mugellini, E.; and Abou Khaled, O. 2020. Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, 179–183. IEEE.

Go, A.; Bhayani, R.; and Huang, L. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford* 1(12):2009.

Huq, M. R.; Ali, A.; and Rahman, A. 2017. Sentiment analysis on twitter data using knn and svm. *International Journal of Advanced Computer Science and Applications* 8(6).

Kouloumpis, E.; Wilson, T.; and Moore, J. 2021. Twitter sentiment analysis: The good the bad and the omg! *Proceedings of the International AAAI Conference on Web and Social Media* 5(1):538–541.

Li, N.; Li, B.; and Gao, L. 2020. Transient stability assessment of power system based on xgboost and factorization machine. *IEEE Access* 8:28403–28414.

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

Nsaif, A. A., and Abd, D. H. 2022. Sentiment analysis of political post classification based on xgboost. In Bashir, A. K.; Fortino, G.; Khanna, A.; and Gupta, D., eds., *Proceedings of International Conference on Computing and Communication Networks*, 177–188. Singapore: Springer Nature Singapore.

Pahwa, B.; S., T.; and Kasliwal, N. 2018. Sentiment analysis- strategy for text pre-processing. *International Journal of Computer Applications* 180:15–18.

Park, M. Y., and Hastie, T. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69(4):659–677.

Qi, Z. 2020. The text classification of theft crime based on tf-idf and xgboost model. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 1241–1246.

Qiu, W. 2019. Credit risk prediction in an imbalanced social lending environment based on xgboost. In *2019 5th International Conference on Big Data and Information Analytics (BigDIA)*, 150–156.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems* 30.

Xu, H.; Liu, B.; Shu, L.; and Yu, P. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2324–2335. Minneapolis, Minnesota: Association for Computational Linguistics.

Yang, L., and Shami, A. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415:295–316.

Zachlod, C.; Samuel, O.; Ochsner, A.; and Werthmüller, S. 2022. Analytics of social media data–state of characteristics and application. *Journal of Business Research* 144:1064–1076.

Appendix

Sentiment Analysis on sentiment140 with XGBoost, BERT, KNN, Naive Bayes, Random Forest, and SVM workflow: https://git.uwaterloo.ca/t4shan/cs-686-sentiment-analysis

URL common start and ends
Start with: http://, https://, www.
End with: com, net, org

Table 8: Appendix 1 URL tokens

Emoji Token	Transferred word
’:)’, ’:-)’	smile
’;d’, ’;)’, ’;-)’	wink
’:-E’	vampire
’:(’, ’:-(’, ’:-¡’	sad
’:P’	raspberry
’:O’	surprised
’:-@’, ’:@’	shocked
’:-$’, ’:-!’, ’:-&’, ’O.o’	confused
’:\\′	annoyed
’:’, ’:X’	mute
’:)̂’, ’:-D’	smile
’$_$’	greedy
’@@’	eyeroll
’:-0’	yell
’(-_-) ’	robot
’d[-_-]b’	dj
”:’-)”	sadsmile
’O:-)’, ’O*-)’	angel
’(:-D’	gossip
’=.̂=̂’	cat

Table 9: Appendix 2 Emoji tokens