Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

Sentiment Classification of Covid-19 Online Articles: The Impact of Changing Parameter Values on Bidirectional LSTM

Samuel Kofi Akpatsa, Hang Lei, Xiaoyu Li, Victor-Hillary Kofi Setornyo Obeng
School of Information and Software Engineering; University of Electronic Science and Technology of China
Chengdu, China

***Abstract*— The advancement of digital technology has played an essential role in shaping the impact of social media. The increasing number of people who use Facebook, Twitter, Instagram, weblogs, and other social networking sites has led to a considerable amount of textual content online, including news articles and historical documents. Information published by online news media has a proactive role in shaping peoples understanding and reactions towards different situations, influencing policy actions. Current research points to a direction where using artificial intelligence techniques enhance text mining capabilities. Stimulated by this, ways to analyze Covid-19 related articles in the context of sentiment classification are sought. This study examines the impact of changing parameter values on the performance of LSTM and BLSTM models for sentiment classification. The dataset used in this study contains more than 10000 Covid-19 online articles published between May 2020 and September 2020. The primary providers of these articles were 10news.com, cnn.com, and foxla.com. The evaluation of the models considered the calculation of accuracy on the validation set. Experimental results show that updating parameters from one experiment to another can gradually optimize the accuracy of deep learning models.**

Keywords*— *NLP, Deep Learning, Text Classification, Sentiment Analysis, Covid-19.

I. INTRODUCTION

The rapid growth in social media usage allowed people of all categories to share their opinions about a wide range of topics. This has led to a considerable amount of textual content online, including news articles and historical documents. Applying text mining methods to discover insights from such corpus can be helpful to individuals, business organizations, and governments. Consequently, Natural Language Processing (NLP) techniques such as sentiment analysis have become an important tool in which attitudes, thoughts, opinions, or judgements towards a particular subject has been extracted. The emergence of the coronavirus pandemic led the mass media providing constant updates on the crisis, describing the impact on the economy and society and the continuous innovations in medical science related to vaccine development and treatments. News articles from well-known online news providers work like a receiver-transmitter sentiment tool that influence policy actions. For example, an economic analyst could rely on information from the media to forecast the economic impact on a business, while a political analyst could write articles to guide people's sentiments.

In the context of data science, the problem of classifying texts according to sentiment can be solved by Sentiment Analysis using predefined polarity assigned to each word. Though easy to apply with modern Python tools like Vader of NLTK and TextBlob, this approach could meet several drawbacks. When using such tools, semantics, morphology, and colloquialisms belong to a non-conceivable spectrum. Moreover, identical words within different language frameworks produce the same polarity, thus skewing what a natural speaker perceives. Figure 1 demonstrates a case of skewed sentiment.

```
1 text = "Good heavens!"
2
3 ('TextBlob:', TextBlob(text).polarity ,
4  'nltk.vader:', vader.polarity_scores(text)['compound'])

('TextBlob:', 0.875, 'nltk.vader:', 0.7088)
```

Figure 1: A case of skewed sentiment

The second solution for a data scientist is to complete this task by constructing a model following the principle "learning from example." This supervised training process aims to estimate the parameters of a problem using training data. Defining the model's structure in advance is the only way for the scientist to use prior knowledge regarding this problem [1].

Recent developments in computer processors allow scientific researchers to use large volumes of text documents, trained on complex ML models with millions of parameters in an acceptable time. However, the difficulty with this kind of approach is that it needs labeled datasets to train ML models to learn over time. For example, to model a binary text classification problem, annotators must first read and assign positive (1) or negative (0) values to each text document. ML models are trained to detect the underlying patterns and relationships between the input text data and the assigned labels. Finally, the trained models can classify accurate labelling results when presented with never-before-seen unlabeled text data

Text classification problems have been studied in many application areas, such as sentiment analysis, topic modeling, recommender systems, and intent detection [2]. One critical challenge confronting most researchers is the abundance of an unlabeled dataset. The volume of data makes it difficult to train learning algorithms in a feasible time which could degrade the accuracy of the classification model. However,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

the advancement of deep learning-based methods has become essential in developing robust and efficient classification models while reducing the training time,

One of the most used text mining techniques in NLP is text feature vectorization. This technique converts input text data into vectors of real numbers, which is the format the ML models support. Prior literature suggests that two of the most common feature vectorization techniques applied in text classification are bag-of-words (BOW) and the Term Frequency - Inverse Document Frequency (TF-IDF) [2]. A more advanced approach such as word embeddings and pre-trained BERT models has been widely used to extract patterns from text sequences. These models have effectively generated language sequences on downstream tasks with higher accuracy, including rich contextual awareness, semantic and syntactic properties, and linear relationships between words.

In modeling text classification problems, the Bayesian model has been effectively applied to calculate the probability of text sequences. Other approaches such as support vector machines, logistic regression, and decision tree algorithms have gained popularity in the efficient implementation of text classification tasks in the past years. The current state-of-the-art for text classification uses DL models to capture complex linguistic information from texts [3]. The non-linearity of DL methods and their ability to easily integrate pre-trained word embeddings often lead to higher classification accuracy than classical linear classifiers [4], [5].

This study demonstrates using a DL method, including layers of Word Embeddings, Convolution, Dropout, and the extension of LSTM to BLSTM, to model covid19 text data. The primary objective is to analyze the impact of extending LSTM to BLSTM in a flow of increasing dropout rates assigned on different layers and achieving progressively higher classification accuracies.

II. PREVIOUS WORK

Sentiment analysis is an active research area in NLP, text mining and web mining, which extract subjective information from source materials. In recent times, various DL models (including DNN, CNN, and RNN) have been used to increase the efficiency of sentiment analysis tasks. This study [6] investigated backdoor attacks against LSTM-based text classification for Sentiment Analysis by poisoning the dataset of IMDB movie reviews. They explain that when one injects the backdoor, the model will misclassify any text samples that contain a specific trigger sentence into a target category. Their experiment achieved a 96% success rate, with a 1% poisoning rate proving that LSTM is also vulnerable to backdoor attacks like CNN.

A related study analyzed the Hierarchical LSTM Network (HAN) for Text Classification [7]. They mined three different datasets and trained CNN, RNN, and HAN for each of these datasets. They concluded that CNN is a decent general method for acceptable validation accuracy, while RNN and HAN did not provide consistent results. Nevertheless, the analysis shows that HAN outperforms other algorithms when treating problems with large-scale datasets.

Another approach used LSTM combined with Word Embeddings to develop Text Classification models for more than 30 languages achieving high accuracy (more than 87%) [8]. They built two axes of binary classifications. The first one was to detect actionability, and the second one was to catch the political preference of tweets. The model was evaluated on an extensive dataset with more than 300,000 rows. Results show that LSTM outperforms other algorithms when combined with Word Embeddings. They also reached a recipe to maximize accuracy that sets LSTM to 32 units, Word Embeddings to 128, and the batch size to 64 observations, adding a Dropout layer before the activation function. Moreover, they investigated the optimizers and the activation function type and concluded that ADAM and the sigmoid function increased accuracy.

Some other research focuses on using a neural network language model on pre-trained word vectors. For example, the authors in [9] utilized the Bidirectional Recurrent Neural Network (BRNN) to substitute pooling layers in CNN to preserve the local information details and catch long-term dependencies. They validated their model using two datasets: (a) Stanford Large Movie Review (IMDB) and (b) Stanford Sentiment Treebank (SST). They concluded that CNN could extract higher-level features invariant to local translation, and the RNN preserved order information even with one layer. They suggested that combining the two models could yield better classification results.

Most approaches that use ensemble learning models toward sentiment analysis involve feature engineering to enhance predictive performance. The authors in this study [10] developed a paradigm of a multi-objective, optimization-based weighted voting framework to assign appropriate weight values to classification algorithms, with each output class based on the predictive performance of the classifiers. The proposed ensemble model is based on static classifier selection involving majority voting error and forward search and a multi-objective differential evolution algorithm. The experimental analysis of classification tasks suggests that the proposed framework outperforms other conventional ensemble learning models such as AdaBoost, bagging, random subspace, and majority voting.

A related study presents an efficient sentiment classification scheme with high predictive performance to analyze a corpus containing 66,000 Massive Open Online Courses Reviews using conventional classification algorithms, ensemble learning, and deep learning methods [11]. The machine learning and ensemble learning methods use weighted feature extraction schemes such as TP, TF, and TF-IDF. The deep learning-based approach utilized word embeddings schemes such as word2vec, GloVe, and FastText for text representation. The empirical analysis indicates that deep learning-based architectures outperform ensemble learning and machine learning methods for sentiment analysis on educational data mining.

Another study used conventional text representation schemes and machine learning classifiers to analyze a corpus containing approximately 700 students' reviews written in Turkish [12]. The empirical results indicate that the machine

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

learning-based approach yields promising results on students' evaluation of higher educational institutions.

This study proposed an effective deep learning-based approach to sentiment analysis on product reviews obtained from Twitter. The presented architecture combines TF-IDF weighted Glove word embedding with CNN-LSTM architecture [13]. The CNN-LSTM architecture consists of five layers; weighted embedding layer, convolution layer, max-pooling layer, followed by LSTM, and dense layer. In the empirical analysis, the predictive performance of different word embedding schemes with several weighted functions has been evaluated in conjunction with conventional deep neural network architectures. The empirical results indicate that the proposed deep learning architecture outperforms the conventional deep learning methods.

III. BACKGROUND

A. *Textual Data*

a. *Handling Textual Data*

Analyzing text documents generally requires some fundamental cleaning and pre-processing steps to improve the quality of the dataset. Text pre-processing includes cleaning text data to remove unnecessary noise such as whitespaces, stopwords, punctuations, misspelt words, and special characters. Words may also be lemmatized or stemmed using well-known and tested automated tools like Wordnet Lemmatizer or Porter Stemmer, both implemented within NLTK. The purpose of these steps is to remove from the text possible features semantically indifferent, extract features (tokens) like words, emoticons, periods, and normalize terms by reducing the morphological index to lemma or stem. Algorithms may use detailed dictionaries to draw the appropriate information [14]. After completing these steps, the remaining entities become the features of the classification problem. The presence and the interaction of these features describe the problem's classes [15].

b. *Vectorizing Text Features*

Vectorization is a feature extraction technique that assigns text features a numerical vector representation that a classification algorithm supports. A good feature vectorization technique enhances the readability of the classification model, help reduce the time required to train a learning algorithm, and increases the model's generalisation ability while eliminating overfitting. The most commonly used BOW illustrates such a vectorizing process creating a sparse representation despite not preserving the sentences' structure [16]. Besides, previous research indicates that integrating pre-train word embeddings with DL models result in superior classification results [5]. The word embeddings representation uses dense vectors to project each word into a continuous vector space. The word's position in the vector space depends on its surroundings. The embedding is the word's position in the learned vector space [17]. Word embeddings create vectors that keep semantic similarity according to the linguistical distributional hypothesis [18].

B. *Layers of Deep Learning Method*

In this study, the chosen DL method is the classifier pipeline (Sequential model) after vectorizing text. In the context of the Keras implementation for DL in Python: "the sequential model allows for a simple stack of layers where each layer has exactly one input tensor and one output tensor" (https://keras.io/guides/sequential_model/)

a. *Convolutional Layer*

A convolution layer as part of a DL method is an automated system for feature extraction from a fixed-length segment. The location of the feature is not significant [19]. There are three hyperparameters to tune: (i) the depth to define the number of neurons to be connected, (ii) the stride to determine the allocation of the depth columns, and (iii) the padding to control the spatial size of the output. A convolution layer on top of a word vector before a fully connected layer can achieve excellent sentence classification results [20].

b. *Max Pooling*

The Max Pooling layer divides the vector into equal rectangles and then outputs the maximum from every rectangle. The intuition behind this process is that the relativity of the location of a feature is critical. The result of this layer is to control overfitting by reducing the spatial size of the representation. Successive convolution layers activated by a Rectified Linear Unit (ReLU) followed by a Max Pooling Layer may prove more efficient [19].

c. *LSTM Layer*

LSTM is an improvement in the context of the standard RNN to address the problem of vanishing and exploding gradients after backpropagation [21]. LSTM units consist of cells and gates. A cell is the memory of the LSTM unit to record the dependencies of the input sequence. With LSTM units, when error values are back-propagated from the output layer, the error remains in the LSTM unit's cell. There are three gates in every unit to control the information flow: (i) the input gate to control the flow of new values, (ii) the output gate to determine whether the values in memory will be used in the activation, and (iii) the forget gate to remove a value from the cell. The activation can be a sigmoid or a tangent function.

d. *Bidirectional Extension on LSTM Layer*

Extending an LSTM layer to Bidirectional connects two hidden layers of opposite directions. With this extension, the output layer simultaneously carries information from the past and the future states [1]. The earlier hidden states only observe a few vectors from the lower layer, while the latter is computed based on most lower-layer vectors [9].

e. *Dropout Layer*

Dropout in the context of a DL method is a regularization process of removing a percentage of nodes from the NN to avoid overfitting and maximizing the model's generalization. The removal of the nodes during dropout is a stochastic process. The reduced network keeps updating its weights, while the removed nodes preserve their weights during this

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

training stage. The results are (a) accelerated training, (b) feature robustness, and (c) depreciation of nodes' interaction [22].

f. Dense Layer

A Dense layer is a fully connected layer where every input is connected to an output with a weight. The Dense layer performs a linear operation, which may end with a non-linear activation function. Several Dense layers may exist in a DL method. However, the last layer in the DL classification tasks calculates the class's probability, and this is a Dense layer with the appropriate non-linear activation. The last layer's usual activation is the sigmoid function [9], but any function can be used for the previous dense layers.

g. Optimization Algorithm

There are several optimization algorithms one may select. Studies have shown that the selection can make a difference in the training time and the result to achieve. For example, a study introduced an efficient stochastic optimization that only requires first-order gradients and little memory requirement [23]. Their method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. They named their algorithm ADAM derived from adaptive moment estimation. Their method combines the advantages of two popular methods: AdaGrad [24], which works well with sparse gradients, and RMSProp [25], which works well in online and non-stationary settings. Studies have compared famous Stochastic Gradient Descent Algorithms and suggested that ADAM might be the overall best choice [8], [26]. Similarly, an experiment on IMDB BoW features suggests that Adam can be more efficient for text classification tasks [23] (Figure 5).

IV. MATERIALS AND METHODS

a. Data Collection and Preparation

We scrape new public data from famous online news providers using keywords like coronavirus and covid-19. A scraper downloaded over 10000 articles corresponding to the date range, May 2020 until September 2020. The primary providers of these articles were *10news.com*, *cnn.com*, and *foxla.com*. The downloaded articles varied significantly in the number of sentences and the word count.

The downloaded articles varied significantly in the number of sentences and word counts. To normalize the text size, we reconstruct the news articles to create paragraphs with approximately ten sentences to utilize as many data points as possible. A script that automatically applies NLTK's sentence tokenizer was run to extract 41839 text entities labeled into different sentiment categories. The NLTK tokenizer was updated to recognize abbreviations like 'Dr.', 'Mr.', 'Mrs.', 'prof.', 'inc.', and 'i.e.' to avoid some known mistakes.

The task of labeling data was handled manually by human annotation. The annotators assigned the value 1 (positive class) for texts with positive sentiment and 0 (negative class) for texts with negative sentiment. After the end of the labeling process, about 8,500 texts were removed from the dataset. Annotators assigned these texts with contradictory labels because of neutrality or interpretation difficulty. The final dataset contained 33,324 texts, of which 62.51% belonged to the positive class, and 37.49% belonged to the negative class [27].

b. Text Cleaning and Preprocessing

The articles contained various features that added a non-necessary burden on dimensionality. Moreover, names of famous personalities like 'Donald Trump' and other entities could add bias to the model. It would be easy for a DL model to learn the names and titles connected to a sentiment instead of learning unbiased text features. There were two cleaning axes: (i) clean tokens semantically insignificant and (ii) clean tokens that may convey bias. Thus, titled names, uppercased words, tokens containing non-alphanumeric characters, numbers, stopwords, and words like 'corona', 'coronavirus', 'covid19' were removed from the text.

This study also introduces an abstraction token to replace 22 negations like not, wouldn't, cannot, can't, mightn't, etc. under the token <negation>. Keeping a negation indicator in the text might help the model to distinguish bigrams and trigrams like "not bad", "not good enough", "cannot do". The remaining tokens were stemmed using the Porter Stemmer of NLTK. The average number of tokens for texts remaining in the dataset was 91, with a maximum of 271 and 75% of the instances to have more than 73 tokens **(Figure 2)**. The tokens number follows the Normal Distribution closely except for a minimal number of outliers creating a right tail. The analysis led to the decision to normalize the data further and remove possible outliers. Under this concept, texts containing less than 36 tokens were removed. This token number corresponds to the 0.0150st quantile of the token's distribution. By this, 1.5% of the data, possibly corresponding to outliers, was excluded.

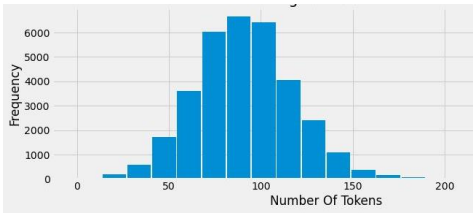

Figure 2: Histogram of token numbers

c. Vectorizing Tokens

The available data is 32,841 sequences of cleaned and stemmed text tokens. The vocabulary of this data reached 22,073 unique tokens. The frequencies of these tokens were calculated and ranked. Also, the minimum acceptable frequency was set intuitively to eight. The most frequent tokens were the: <said> with 50,237 appearances, <negation> with 28,093, <people> with 22,064, <state> 16,358, and <case> with 15,676. The token <negation> corresponds to stopwords preserved in the text under this abstraction token as described in section 4.2. Some tokens with minimum frequency were <swoosh>, <investiture>, <unclimb>, <calfir>, <mandarin>.

For constructing the sequences, unique integers replaced every token. Tokens having frequency under the minimum (8)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

received zero value. The length of every sequence was set to the 0.9850st quantile of the token distribution (152). Thus, after truncating and padding the sequences, the vectorized dataset consisted of 32,841 sequences of 152 units.

d. Train/Test Split

In this study, 67% of the data was used as the training set and the remaining 33% as the validation set. The split was random, but a seed was selected for reproducibility. The number of sequences used to train the model was 22,003, and the number of sequences to test it was 10,838.

e. Sequential Model

The building strategy of the sequential model was to add a dropout after every layer. Conducting several experiments with fluctuating parameters and taking the maximum accuracy after training models differing to the Bidirectional extension of LSTM would create a flow of comparable accuracies.

The first layer on the stack would be an Embedding layer with input the vocabulary number (22,073 unique tokens) and output a 64-dimension vector of length 152 (0.9850st quantile of the token distribution). A Dropout layer would follow. The third layer would be a one-dimensional convolution layer with 16 filters and an activation function set to ReLU, followed by a Max Pooling layer with pool size set to two. The fifth layer would again be a Dropout layer, and then the LSTM layer would follow either with a Bidirectional extension or not. The LSTM layer also supports internal dropouts, but one more Dropout layer corresponds to the whole system. The system is completed with a Dense layer with a non-linear activation (sigmoid function) to calculate the final output (probabilities), a binary cross-entropy as the loss function, and ADAM's declaration be the DL optimizer. Table 1 summarizes the layers and the parameters of the model.

Table 1: LSTM Baseline (Model: 'sequential_132')

Layer type	Output shape	Param #
embedding_132 (Embedding)	(None, 152, 64)	1412736
dropout_139 (Dropout)	(None, 152, 64)	0
conv1d_131 (Conv1D)	(None, 152, 16)	3088
max_pooling1d_131 (MaxPooling)	(None, 76, 16)	0
dropout_393 (Dropout)	(None, 76, 16)	0
lstm_131 (LSTM)	(None, 32)	6272
dropout_394 (Dropout)	(None, 32)	0
dense_131 (Dense)	(None, 1)	33

f. Impact of Bidirectional Extension

A way to analyze how BLSTM may impact optimal parameters is to conduct six experiments, training LSTM and BLSTM models for five (5) epochs using the train set. Every experiment regards changing the values of one parameter. This concept uses the validation data to calculate the accuracy score and record it for every epoch. After completing the training for every parameter value, a comparison of the achieved maximum accuracy for every value demonstrates the Bidirectional extension's impact on maximum accuracy. Updating the parameters from one experiment to the other gradually optimizes accuracy, performing and proposing a parameter tuning for both LSTM and BLSTM. The evaluation of the results proposes ways to handle parameters to utilize the Bidirectional extension of LSTM.

V. RESULT

a. Batch Size

The first experiment demonstrated how the bidirectional extension could influence the optimal batch size. There were 6 sizes tested: [16, 32, 64, 128, 256, 512]. **Table 2** illustrates the maximum accuracies achieved for every batch size. LSTM and BLSTM seem to provide higher accuracies for smaller batch sizes. BLSTM seems to be more stable than LSTM despite achieving lower accuracies. The highest LSTM accuracy was 0.8966 for batch size 32 versus 0.8951 for batch size 16 for BLSTM.

Table 2: Validation accuracy vs Batch Size

Batch size	LSTM	BLSTM
16	0.8939	0.8951
32	0.8966	0.8927
64	0.8909	0.8931
128	0.8955	0.8891
256	0.8882	0.8885
512	0.8867	0.8869

b. Dropout After Embedding Layer

The second experiment analyzed the dropout after the embedding layer. The embedding layer provides 64 dimensions and four (4) dropout values were tested: [0.10, 0.25, 0.50, 0.75]. The batch size for this iteration was set to 64 for both LSTM and BLSTM. The maximum accuracy for LSTM reached 0.8977 for a 0.50 dropout and BLSTM 0.8960 for a 0.25 dropout. It seems that a dropout rate after embeddings can be useful for both LSTM and BLSTM. BLSTM does not seem to perform better than LSTM, yet. Figure 7 shows these results, and one may also observe that BLSTM is again more stable than LSTM.

Table 3: Dropout after Embedding Layer

Dropout Values	LSTM	BLSTM
0.10	0.8935	0.8918
0.25	0.8961	0.8960
0.50	0.8977	0.8940
0.75	0.8915	0.8946

c. Dropout After Convolution Layer

The third experiment analyzed the dropout after the convolution and the max-pooling layer. The dropout values tested were the: [0, 0.10, 0.25, 0.375, 0.50, 0.625, 0.750]. The batch size and the dropout after embedding were set to 64 and 0.50, respectively, for LSTM and BLSTM. The best result for LSTM was a maximum accuracy of 0.8967 achieved for a 0.10 dropout. On the contrary, BLSTM achieved a 0.9008 accuracy for a 0.25 dropout. The results from **Table 4** suggest that tuning the dropout rate after convolution and max-pooling may prove more important when extending LSTM to Bidirectional.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

Table 4: Dropout rate after Convolution Layer

Batch size	LSTM	BLSTM
0.000	0.8932	0.8921
0.100	0.8980	0.9032
0.250	0.8982	0.8968
0.375	0.9007	0.8919
0.500	0.8893	0.8896
0.625	0.8584	0.8890
0.750	0.8336	0.8818

d. LSTM Internal Non-Recurrent Dropout

The LSTM layer provides two internal dropout parameters: a recurrent and non-recurrent dropout. Internal Non-Recurrent Dropout refers to the fraction of the units to drop for the inputs' linear transformation. The fourth experiment of this study tested how Bidirectional extension may influence maximum accuracy in a flow of different dropout rates than a simple LSTM after performing some parameter searching. The batch size and the dropout after embedding did not change. Dropout after the convolution layer is set to 0.10 for LSTM and 0.25 for BLSTM. The experiment resulted in LSTM achieving a 0.9007 maximum accuracy for a 0.375 dropout rate and BLSTM a 0.9032 for a 0.10 dropout rate (**Table 5**). Both LSTM and BLSTM appeared to earn a bit of accuracy. Besides, the previous observation about BLSTM being more stable but demanding careful parameter tuning was also confirmed

Table 5: Validation accuracy after LSTM Dropout

Batch size	LSTM	BLSTM
0.000	0.8918	0.8935
0.100	0.8968	0.8911
0.250	0.8939	0.9008
0.375	0.8967	0.8968
0.500	0.8902	0.8950
0.625	0.8968	0.8955
0.750	0.8930	0.8924

e. LSTM Recurrent Dropout

In this study, recurrent dropout is the fraction of the units to drop for the linear transformation of the recurrent state. The fifth experiment explored the possibility of improving accuracy when training with several recurrent dropout rates. The rates to run are: [0, 0.10, 0.25, 0.375, 0.50, 0.625, 0.75] while the parameters of non-recurrent dropout were updated to 0.375 for LSTM and 0.10 for BLSTM. This experiment indicates that recurrent dropout in the LSTM cell limits the amount of information the model receives. Nevertheless, BLSTM accuracy seems steadily higher than LSTM (**Table 6**).

Table 6: Validation Accuracy after Recurrent Dropout

Batch size	LSTM	BLSTM
0.000	0.8920	0.8978
0.100	0.8911	0.8690
0.250	0.8684	0.8912
0.375	0.8593	0.8915
0.500	0.8679	0.8844
0.625	0.8630	0.8750
0.750	0.8643	0.8806

f. Final Dropout

The previous experiment suggests that recurrent dropout in the LSTM cell cannot improve the accuracy of either the LSTM or the BLSTM models. For validating our results until this stage, a final experiment was run with dropout rates: [0, 0.10, 0.25, 0.375, 0.50, 0.625, 0.75]. We expected that a non-zero rate could not beat the previous highest accuracies. A different result would mean that dropout rates could further receive updates with greater values. The highest accuracy for BLSTM was 0.9002, achieved for a zero rate, while the highest accuracy for LSTM was 0.8980, achieved with a 0.10 rate (**Table 7**).

Table 7: Validation Accuracy after the Final Dropout

Batch size	LSTM	BLSTM
0.000	0.8979	0.9002
0.100	0.8980	0.8922
0.250	0.8892	0.8980
0.375	0.8963	0.8981
0.500	0.8921	0.8935
0.625	0.8832	0.8989
0.750	0.8969	0.8955

VI. FUTURE WORK

The contribution of DL and Text Preprocessing in achieving a classification accuracy greater than 0.90 could support additional research. In the Text Preprocessing section, we introduced the abstraction token. However, it was out of this study's scope to analyze it further and discover its possible contribution to the accuracy. We strongly believe that finding ways to increase abstraction features in a model may utilize new entities. In the future, we wish to measure the abstraction's token contribution to the accuracy of a classification model in sentiment analysis and extend it to lexical entities different from stopwords.

Moreso, this study achieved a good classification accuracy while analyzing the impact of the Bidirectional extension of LSTM during a Dropout fluctuation. Further work may perform a similar analysis on changing LSTM units or Embedding units that have been kept steady during this experiment.

VII. CONCLUSION

This study used a labeled dataset to analyze the impact of the Bidirectional extension of the LSTM layer in sentiment analysis. We conducted five experiments to analyze the connection of dropout rates to the maximum accuracy the classifier can achieve and to analyze the batch size. The findings may support the notion that BLSTM can outperform LSTM when tuned carefully and with detail. The experiments also demonstrated that while BLSTM needs meticulous tuning to maximize its predictability, it can also handle inaccurate and unlucky parameter tuning more successfully than LSTM. We additionally indicated that LSTM efficiency could vary, and randomness may play a role and should be considered when training a model.

We also analyzed the idea of combining Dropout layers after Embeddings, Convolution, and LSTM layers, and we found no proof against this. The same idea also suggests that the final Dropout layer may be of no use if the previous ones are combined successfully. The first experiment (5.1) also confirmed the idea that a smaller batch size can achieve higher accuracy. Finally, we proved that sentiment analysis using DL could be successful for text documents characterized by formality and objectivity such as those from prominent news

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

organizations with the primary purpose of informing rather than expressing personal and subjective opinions and judgments

DATA AVAILABILITY

The dataset is available at Mendeley Repository: https://data.mendeley.com/datasets/r6nn5s37tp/2

ACKNOWLEDGMENT

This study was supported by the National Key R\&D Program of China, Grant No. 2018YFA0306703.

CONFLICTS OF INTEREST

The authors declare that the research was conducted without any commercial or financial relationships that could be interpreted as a potential conflict of interest.

REFERENCES

[1] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[2] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[3] S. K. Akpatsa, X. Li, and H. Lei, "A Survey and Future Perspectives of Hybrid Deep Learning Models for Text Classification," in *International Conference on Artificial Intelligence and Security*, 2021, pp. 358–369.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] Y. Goldberg, "A primer on neural network models for natural language processing," *J. Artif. Intell. Res.*, vol. 57, pp. 345–420, 2016.

[6] J. Dai, C. Chen, and Y. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, vol. 7, pp. 138872–138878, 2019.

[7] K. Borna and R. Ghanbari, "Hierarchical LSTM network for text classification," *SN Appl. Sci.*, vol. 1, no. 9, p. 1124, 2019.

[8] A. Rao and N. Spasojevic, "Actionable and political text classification using word embeddings and lstm," *arXiv Prepr. arXiv1607.02501*, 2016.

[9] A. Hassan and A. Mahmood, "Efficient deep learning model for text classification based on recurrent and convolutional layers," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017, pp. 1108–1113.

[10] A. Onan, S. Koruko\uglu, and H. Bulut, "A multi-objective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Syst. Appl.*, vol. 62, pp. 1–16, 2016.

[11] A. ONAN, "Sentiment analysis on massive open online course evaluations: a text mining and deep learning approach," *Comput. Appl. Eng. Educ.*, vol. 29, no. 3, pp. 572–589, 2021.

[12] M. A. Toço\uglu and A. Onan, "Sentiment analysis on students' evaluation of higher educational institutions," in *International Conference on Intelligent and Fuzzy Systems*, 2020, pp. 1693–1700.

[13] A. Onan, "Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks," *Concurr. Comput. Pract. Exp.*, p. e5909, 2020.

[14] E. Leopold and J. Kindermann, "Text categorization with support vector machines. How to represent texts in input space?," *Mach. Learn.*, vol. 46, no. 1–3, pp. 423–444, 2002.

[15] E.-H. S. Han, G. Karypis, and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification," in *Pacific-asia conference on knowledge discovery and data mining*, 2001, pp. 53–65.

[16] X. Hu and H. Liu, "Text analytics in social media," in *Mining text data*, Springer, 2012, pp. 385–414.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[18] H. Widdowson, "JR Firth, 1957, Papers in Linguistics 1934--51," *Int. J. Appl. Linguist.*, vol. 17, no. 3, pp. 402–413, 2007.

[19] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[20] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv Prepr. arXiv1408.5882*, 2014.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv Prepr. arXiv1412.6980*, 2014.

[24] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *J. Mach. Learn. Res.*, vol. 12, no. 7, 2011.

[25] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *Univ. Toronto, Tech. Rep.*, 2012.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv Prepr. arXiv1609.04747*, 2016.

[27] S. K. Akpatsa, "Covid-19 Online Article Dataset," *Mendeley Data doi 10.17632/r6nn5s37tp.2*, 2021.