WRS: A Novel Word-embedding Method for Real-time Sentiment with Integrated LSTM-CNN Model

Conference Paper · July 2021

DOI: 10.1109/RCAR52367.2021.9517671

4 authors, including:

Abdur Rasool
Chinese Academy of Sciences
20 PUBLICATIONS **66** CITATIONS
SEE PROFILE

Qingshan Jiang
Chinese Academy of Sciences
201 PUBLICATIONS **2,688** CITATIONS
SEE PROFILE

Proceedings of The 2021 IEEE International
Conference on Real-time Computing and Robotics
July 15-19, 2021, Xining, China

WRS: A Novel Word-embedding Method for Real-time Sentiment with Integrated LSTM-CNN Model

Abdur Rasool[1,2], Qingshan Jiang[1,*], Qiang Qu[1], Chaojie Ji[3]

2021 IEEE International Conference on Real-time Computing and Robotics (RCAR) | 978-1-6654-3678-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/RCAR52367.2021.9517671

***Abstract*—Artificial Intelligence (AI) is a research-focused technology in which Natural Language Processing (NLP) is a core technology in AI. Sentiment Analysis (SA) aims to extract and classify the people's opinions by NLP. The Machine Learning (ML) and lexicon dictionaries have limited competency to efficiently analyze massive live media data. Recently, deep learning methods significantly enrich the accuracy of recent sentiment models. However, the existing methods provide the aspect-based extraction that reduces individual word accuracy if a sentence does not follow the aspect information in real-time. Therefore, this paper proposes a novel word embedding method for the real-time sentiment (WRS) for word representation. The WRS's novelty is a novel word embedding method, namely, Word-to-Word Graph (W2WG) embedding that utilizes the Word2Vec approach. The WRS method assembles the different lexicon resources to employ the W2WG embedding method to achieve the word feature vector. Robust neural networks leverage these features by integrating LSTM and CNN to improve sentiment classification performance. LSTM is utilized to store the word sequence information for the effective real-time SA, and CNN is applied to extract the leading text features for sentiment classification. The experiments are conducted on Twitter and IMDB datasets. The results demonstrate our proposed method's effectiveness for real-time sentiment classification.**

I. Introduction

In recent years of big data, text data is becoming massive due to various data generation platforms. For instance, 350 million online Twitter users are posting 500 million tweets every day. These tweets/texts deliver their emotions or views about real-time events or activity. The best practice to detect such people's reactions from the tweets is to exercise the Sentiment Analysis (SA) techniques, which deal with opinion mining, NLP, and polarity classification [1]. SA has been applied in various real-world problems such as prediction of the stock exchange market and brand building. It also has a huge influence on political and government issues by analyzing people's thoughts for the necessary decision and policymaking [2]. Rajasree R. et al. proposed a method to analyze the tweets for electronic products such as laptops etc by offering a novel feature vector and found that Naïve Bayes (NB) delivers better accuracy [3]. Similarly, various other studies [2] [9] [22] yielded a good SA performance with machine learning algorithms and lexicon dictionaries. These hybrid models for sentiment provide adequate accuracy for particular domains. However, it renders the overfitting issue that leads to low accuracy with other domains when taking real-time data. Meanwhile, the ML algorithm requires score-based (numeric) data to detect meaningful knowledge from text data.

A study [4] suggested employing the word embedding methods to overcome these issues. The word embedding approach is a distribution representation of text into a numeric format that maps the individual word from a document to a pre-defined vector space. A word embedding technique, Word2Vec, detects the similarity scores from the trained corpus based on a vector space's window size [5]. It can predict a target word given the surrounding or context words (set of words at a specific distance before and after that particular word in the corpus). Andi R. et al. presented an approach for the automatic prediction of polarity with Word2Vec embedding. It conducted the experiment on NB and Support Vector Machine (SVM) for the feature representation and achieved a 90% F-score with the Word2Vec technique [6]. The Word2Vec method is the best approach for medium and large data sizes [7].

As our work is considering a massive amount of text data, we employed Word2Vec for our method. We proposed a word-to-word graph (W2WG) embedding to capture the words' co-occurrence information between the words. The fundamental concept for graph-based word embedding is to design such word feature graphs which embed the keywords into nodes and edges of a graph. These features capture the global structure information from the word co-occurrence graph. However, it is still critical to detect the impressive performance for real-time sentiment classification. To improve the feature competency, we have utilized two different neural networks, LSTM (Long Short Term Memory) and CNN (Convolutional Neural Network), by integrating them.

LSTM [8] provides the solution for learning long-term dependencies. It introduces the memory cells that can help to hold the feature for a longer time and evaluate the features more precisely. Recently, LSTM has been widely used in NLP tasks, particularly in sentiment and opinion mining [9] [10]. Similarly, CNN can process the data along with n-gram features which is more capable for various tasks of NLP. Each layer of the CNN model produces more context features for context representation. The features captured from all context representation are used to detect the sentiment polarity [11]. The author [12] proposed a solution to solve the complex context and data pre-processing parallel using CNN. It extracts the features for the final sentiment classification.

This research work is supported by The National Key Research and Development Program of China under Grant No. 2020YFA0909100, and National Natural Science Foundation of China under Grant No. 61902385.

[1]Shenzhen Key Lab for High Performance Data Mining, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (Corresponding author, e-mail: qs.jiang@siat.ac.cn).

[2]Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: rasool@siat.ac.cn).

[3]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China (e-mail: cj.ji@siat.ac.cn).

In this paper, we proposed the WRS method that deals with a Word embedding method for the Real-time Sentiment for word representation. It utilizes novel word embedding methods for the extraction of sentiment knowledge. Firstly, various ways are used to assemble the different lexicon resources for lexical sentiment classification. Secondly, the W2WG embedding that is the first novelty of WRS is performed using the Word2Vec techniques. Then, it adopted the pre-defined cosine distance to compute the similarity between the words and utilized the WordRank algorithm for the word ranking problem of Word2Vec embedding. The novel embedding method enables WRS to extract feature vectors. Then, two different neural networks have been employed for the feature representation. The LSTM and CNN have been integrated to achieve effective sentiment classification. We added the fully connected layer and softmax classifier into the integrated network for better classification results. The integration strategy will enable WRS to achieve the intrinsic features, which will assist in gaining a higher accuracy rate. For the experiments, we used two datasets for the classification; a well-known film review text data [13] from the Internet Movie Database (IMDB), and we crawled real-time data from the microblogging site, Twitter, for COIVD-19 based tweets. The significant contributions of this article are as follows:

- We propose a novel Word-to-Word Graph (W2WG) embedding to the real-time sentiment (WRS) for the word representation by assembling different lexicon dictionaries.
- Two popular neural networks integrate (LSTM-CNN) to gain the highly efficient feature for real-time sentiment analysis.
- We experiment with IMDB and real-time Twitter data on our method and achieved effective results.

The rest of the article is arranged as follows: Section II describes the preliminary review. Section III presents the proposed WRS method. Section IV delivers the experiments and results, and the conclusion is in Section V.

II. PRELIMINARIES

This section presents the introduction of those pre-defined technologies which other authors have delivered.

A. Assembling of Lexicon Dictionaries

There is an enormous variety of text due to active users from various locations. The available lexicon dictionaries have different volumes of words. If a specific word is not existing in it, the sentiment word will be ignored. Thus, we designed an assemble method of lexicon resources to enhance individual dictionary capability. The further specifics of these lexicon dictionaries are shown in Table [1].

TABLE I. AN OVERVIEW OF DIFFERENT LEXICON RESOURCES

Sentiment lexicons	No. of words	Classification	Score
SentiWordNet [14]	155,287	Positive, Negative	[0, 1]
Sentiment140 [15]	62,468	Positive, Negative	[-7, +7]
SenticNet 4 [16]	50,000	Positive, Negative	[-1, +1]

B. Word Embedding

One of the existing word embedding methods, Word2Vec, is learned on the training corpus. Word embedding's graph offers two different kinds of information; (i) local contexts and (ii) weight of co-occurrence among words that are vital for specific negative keywords extraction. The first kind of information delivers local correlation patterns among words, which is difficult for re-raking the words. The second kind of information delivers the edge weight ω that is associated with co-occurrence frequency and distances between two nodes; w_i $(i = 1, 2, \dots n)$ and w_j $(j = 1, 2, \dots n)$. Hence, edge weight w_{ij} ϵ ω between two words is defined as [17]:

$$w_{ij} = \sum_{k=1}^{fr(w_i, w_j)} \frac{1}{l_k(w_i, w_j)} \quad (1)$$

where, $fr(w_i, w_j)$ is the frequency of co-occurrence, while, $l_k(w_i, w_j)$ is a list of words between w_i and w_j with k_{th} co-occurrence for specified window size.

C. Computation of Cosine Distance

As the Word2Vec is measured by the word vector scores. These scores are calculated by computation of Cosine distance D_c between two words w_i and w_j, and a different vector dimension. The angle θ is defined between w_i and w_j. This metric [18] is defined as:

$$D_c(w_i, w_j) = 1 - cos(\theta)$$

$$D_c(w_i, w_j) = 1 - D_s(w_i, w_j) \quad (2)$$

where, $D_s(w_i, w_j)$ denotes the cosine similarity that is relevantly used in text mining to compare the documents. It represents the cosine distance [18] that defined as:

$$D_s(w_i, w_j) = \frac{\sum_{i=1}^{n} w_i \cdot \sum_{j=1}^{n} w_j}{\sqrt{\sum_{i=1}^{n} w_i^2} \sqrt{\sum_{j=1}^{n} w_j^2}} \quad (3)$$

This cosine similarity to achieve the maximum similar vectors between two words w_i and w_j.

D. LSTM

In a deep learning neural network, a long short-term memory (LSTM) networks are employed to avoids the RNN (Recurrent Neural Network) problem efficiently. LSTM network has three gates embedded with two layers; input gate i_t, forget gate f_t, and output gate o_t, while hidden layer h_t and memory storage layer c_t. The input gate controller deals with the new information either to be stored or not as input. The forget gate makes assure how long specific information can be stored in memory. In contrast, the output gate determines the effects of stored information on output activation [10]. Fig. 1 illustrates the LSTM architecture.

The memory storage layer (C_t) deals with the reservation of previous sequence data or information. Equation (4) indicates the memory storage layer's formula; similarly, the formulas of i_t, f_t, and o_t are in (5), (6), and (7), respectively. Wherein the b presents the bias and W shows the weight of each layer, σ symbol indicates the sigmoid function, x_t denotes the input data

of given units, and *tanh* presents the tangent function of the LSTM unit. As a result of output gate information and storage layer, a hidden layer (h_t) is generated for the LSTM unit, as presented in (8) [10].

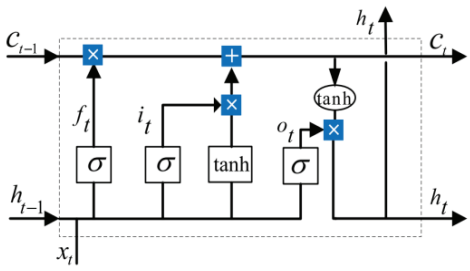

Figure 1. A classical LSTM architecture [10].

$$C_t = i_t \cdot g_t \cdot f_t \cdot C_{t-1} \quad (4)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}; X_t] + b_i) \quad (5)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}; X_t] + b_f) \quad (6)$$

$$g_t = tanh(W_g \cdot [h_{t-1}; X_t] + b_g) \quad (7)$$

$$h_t = O_t \cdot tanh(C_t) \quad (8)$$

$$O_t = \sigma(W_o \cdot [h_{t-1}; X_t] + b_o) \quad (9)$$

The LSTM has an intrinsic ability of information storage, enabling i_t to expand it from a unit to time-series. It has a good capability to process the sequence data, such as a document (in our case, it's a tweet or text-based view). We have adopted the LSTM for the encoding representation of each document.

E. CNN

CNN has four fundamental layers; input layer, convolution layer, pooling, and fully connected layer. The architecture of these four CNN layers is presented in Fig. 2. In the first layer, some significant pre-processing tasks such as word segmentation is processed. In our case, we segment these words by word embedding. Meanwhile, the convolution and pooling layer is vital in our case, processed by (10) and (11), respectively [19].

$$h_j^l = f\left(\sum_{i \in Z}^{i=0} x_i^{l-1} * w_{ij}^l + b_j^l\right) \quad (10)$$

$$c_j = f\left(max_{j \in k} h_j + b_j\right) \quad (11)$$

$$s_\theta(y) = \sum_{c_j \in C}(c_j * \theta_j + b_j) \quad (12)$$

$$P(y = 1|c; \theta) = \frac{e^{s\theta(t)}}{\sum_t e^{s\theta(t)}} \quad (13)$$

The notation in these equations; x_i presents the input features, w_{ij} denotes the kernel weight, Z shows the feature maps, k indicates the pooling windows, and h delivers the input data for the pooling layers. Equation (12) and (13) presents these function where $p(y=t)$ indicates the probability of t, while t shows the time of category sample [19]. In this paper, we have utilized CNN for the text classification, as illustrated in Fig. 2.

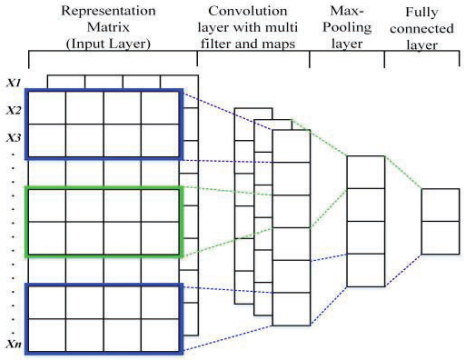

Figure 2. A classical CNN architecture with four layers.

III. METHODOLOGY

An integrated diagram is architected for the proposed WRS method, which focuses on implementation and the fundamental aspects of SA with the word embedding and integrated LSTM-CNN model. The proposed WRS method is based on six different executable tasks, as illustrated in Fig. 3. The details of each step in give as follows:

1. Data Acquisition & NLP Pre-Processing: In this work, two different text datasets have been used; Twitter data and IMDB data. Twitter stream data is crawled by Tweepy API by using the keywords of COVID-19. This data contains the English tweets of December 2020. We applied some vital pre-processing techniques, such as garbage removal, stemming, and POS tagging, to make the data clear for better model performance due to tweets ambiguity. In contrast, IMDB data is publically data that has been collected from [13].

2. Assembling of Lexicon Dictionaries: We assembled the three different lexicon resources for the lexicon sentiment classification. We combined these specific lexicon dictionaries for this purpose. This assembling method for each particular dictionary is the given following:

- SentiWordNet – The following equation is used to compute the sentiment scores within a particular range of [-1, +1]:

  Senti_Score = (PosScore – NegScore)

  where the PosScore is a positive sentiment score, while NegScore is a negative sentiment score with a general rule, i.e., if *senti_score > 0*, then the given word will be positive otherwise negative.

- Sentiment140 – This dictionary presents the different colors to the words, e.i., green to positive, red to negative, and white to neutral words. We assigned the +1 to positive, -1 to negative, and 0 to neutral words.

- SenticNet 4 – This dictionary already has a sentiment classification into positive and negative, while we have assigned the sentiment score [-1, +1] to each word.

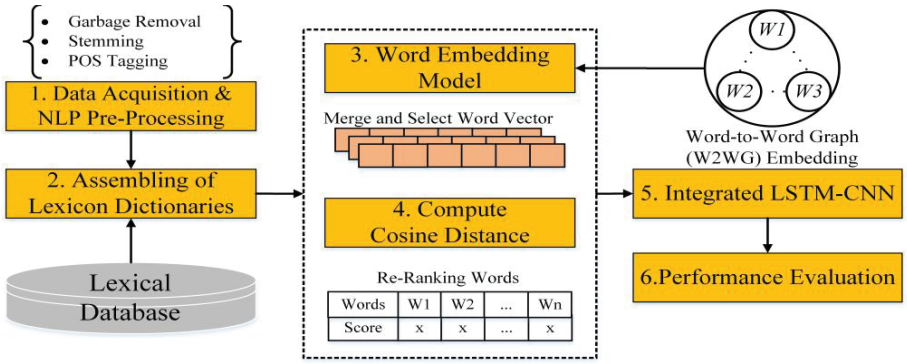

Figure 3. Proposed real-time sentiment method (WRS) with word embedding and integrated LSTM-CNN models.

3. W2WG Embedding: We proposed a novel method of word-to-word graph embedding that digs out the words' co-occurrence information among the words from the given corpus. It is displayed:

$$G_{w_{ij}} = (W, E_{w_{ij}}, \omega_{w_{ij}})$$

where W shows the vocabulary of words, $E_{w_{ij}}$ shown the co-occurrence edges among words, while $\omega_{w_{ij}}$ represents the edge weights. The aim of $G_{w_{ij}}$ embedding is to deliver a low dimensional vector between two words. This aim is achieved by minimizing the Kullback-Leibler Divergence [17] between the empirical probability and joint probability.

$$\tau(W) = \sum_{(i,j)\in E_{w_{ij}}} \check{P}(w_i, w_j)\, log \frac{P(w_i,w_j)}{P(w_i,w_j)} \qquad (14)$$

where the empirical probability $\check{P}(w_i, w_j)$ is computed as:

$$\check{P}(w_i, w_j) = \frac{\omega_{ij}}{\sum_{(i,j)\in E_{w_{ij}}} . \omega_{ij}}$$

While the joint probability $P(w_i, w_j)$ is calculated as:

$$P(w_i, w_j) = \frac{1}{1+exp(-w_i . w_j)} \qquad (15)$$

In the current word embedding approaches, the conditional probability is employed between the target word w_i and context word w_j to predict the target word. We achieved a low dimensional vector between two words by assigning equal values to both words in the same co-occurrence by using the joint probability $P(w_i, w_j)$ (15) to get word embedding vector $\vec{w}$ with low dimensional. The proposed word-to-word graph W2WG embedding is presented in Algorithm 1. It takes three different inputs and performs the equation (15) after computing the negative samples for each word. These words are further combined and reselect for the maximum similar vectors by calculating the cosine distance.

4. Compute Cosine Distance: In this work, we adapted the cosine similarity (3) to attain the maximum similar vectors. The word ranking problem for the Word2Vec embedding is tackled by an efficient WordRank algorithm proposed by [18].

Algorithm 1 Word-to-Word Graph (W2WG) embedding.

Input: (1) Word to word graph for word embedding ($G_{w_{ij}}$); (2) A vocabulary of words $W(w_i, w_j)$ with number of negative samples k; (3) Co-occur edge weight between two words ($\omega_{w_{ij}}$);

Output: Word embedding vector $\vec{w}$;

1: w_i , w_j $\forall$ w $\in W$;
2: **while** $\omega_{w_{ij}} \leq W$ **do**
3: Sample one edge (w_i, w_j) from $G_{w_{ij}}$ and measure ω_{ij} by (1);
4: **For** $k = 0; k < K; k = k + 1$ **do**;
5: Sample a negative word w_j for w_i;
6: Sample a negative word w_i for w_j;
7: Update the W based on (15);
8: **end for**
9: **end while**
10: **return** word embedding vector $\vec{w}$.

5. Integrated LSTM-CNN Model: We have engaged a multi-layer neural network by integrating LSTM [10] and CNN [19] and named LSTM-CNN. It has a simultaneous based advantage of data sequence processing and feature extraction for text classification. The architecture of integrated LSTM-CNN is presented in Fig. 4. After utilizing the proposed W2WG embedding technique, which generates the feature vector, the LSTM unit is adapted to insert the feature vector from one side as an input. LSTM will processes this input and handle the data sequence variable length. It stores the semantic information in its given units and output as a matrix of text representation. In the next step, CNN will utilize this text matrix of LSTM. During the employment of convolution and pooling layers, feature vectors are extracted separately. We operated the fully connected layer to gain the feature values for each class of sentiment. Furthermore, we used the softmax function to compute the sentiment probability. Similarly, a stochastic gradient is adopted to achieve sufficient accuracy by training our model. As a result, the sentiment information is gained by implementing the CNN.

6. Performance Evaluation: The Confusion matrices are keys to assessing the utilized model's efficiency [21]. These matrices, including Precision, Recall, and F-score, are performed to measure performance in this study.

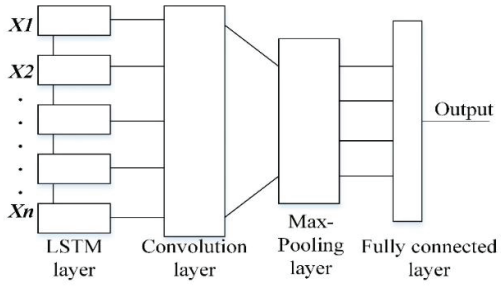

Figure 4. Integrated LSTM-CNN network model for text classification.

IV. Experiments and Result Evaluations

This section elaborates the data description, experiment process, and evaluation of the proposed WRS method's result.

A. Data Description

Our crawler has automatically collected the real-time tweets based on COVID-19 keywords. It contains the one million tweets, which have been pre-processed, and finally, 627,093 tweets have been received as structured data. Similarly, the IMDB dataset has 50,000 reviews of movies [13]. These datasets have been divided into 7:3 proportions for training and testing of our model, respectively. Both datasets have been passed through the lexically based sentiment to distribute the text documents into positive and negative classes. Note that we have not considered the neutral emotion in this study due to its normal behavior. The detailed description of both datasets used in this paper is given in Table 2.

TABLE II. The Statistic Of Twitter And IMDB Dataset.

Corpus	Total dataset	Pre-processed	Positive	Negative	Total
Twitter	1 M	627,093	234,710	168,319	403,029
IMDB	50,000	…	24,290	25,710	50000

B. Implementation Process

After collecting and pre-processing the dataset, we have applied the models on both datasets separately. Firstly, for the W2WG embedding, we adjust some parameters such as Word2Vec size: 300, windows: 7, and epoch 32. We utilized the NLTK library to import its primary functions, such as tokenizer and stopwords. The W2WG provides us a list of feature vector which are sufficiently efficient to be learned for the sentiment classification. However, we have adopted two integrated neural networks for effective classification. For the integrated LSTM-CNN model, we imported their layers from the Keras library, such as convolution, maxpooling, etc. These models enabled the embedding method's feature vector to extract the highly efficient features to be classified with neural networks. We also have used some baseline models of machine learning, such as SVM with Polynomial kernel and Multinomial NB. For which we adopted the sklearn library to implement these classifiers. We have conducted all these experiments on TensorFlow by using Python language. Our system specs are Windows 10 operating systems with Intel i7-9750H CPU.

C. Model Performance's Evaluation

We have implemented different ML and neural network models with our proposed method, WRS. Tables 3 and 4 provide the output values of these models with the Confusion matrices. The F-score is a weighted average of precision and recall, which gives the performed accuracy of a given model; the higher the value, the higher the accuracy.

The evaluation of Table 3 presents the model's performance with the real-time Twitter data set. The traditional machine learning algorithm's performance is averagely lower than the neural networks, such as SVM and NB gaining 0.75% and 0.71% scores, respectively. The Word2Vec with W2WG embedding performed effectively than the ML models, LSTM and CNN individually. However, with the integration of LSTM-CNN, our proposed method boosts its F-score=0.881% performance for the real-time sentiment classification.

TABLE III. The Model's Performance Evaluation With Our Proposed WRS Method On Twitter Dataset.

Models	Precision	Recall	F-score	Execution Time (s)
Polynomial SVM	0.74	0.78	0.759	378.6
Multinomial NB	0.71	0.73	0.719	410.3
Word2Vec	0.82	0.81	0.814	629.7
LSTM	0.69	0.74	0.714	1064.1
CNN	0.79	0.82	0.804	297.9
LSTM-CNN	**0.87**	**0.88**	**0.881**	**449.3**

The evaluation of the IMDB dataset has been provided in Table 4. It also indicates that average machine learning models' efficiency with the given method is lowest than a deep neural network. However, when we implement our proposed method with the combinational neural networks, the model's accuracy exceeds up to 0.898%. In contrast, it is crystal clear that the model accuracies on IMDB datasets outperformed the Twitter dataset. The possible reason for the best performance on IMDB data can be the highly structured and organized dataset. In the case of the Twitter dataset, the model's accuracy can be improved by considering more pre-processing techniques to make data smoot and highly efficient.

TABLE IV. The Model's Performance Evaluation With Our Proposed WRS Method On IMDB Dataset.

Models	Precision	Recall	F-score	Execution Time (s)
Polynomial SVM	0.80	0.82	0.809	216.4
Multinomial NB	0.68	0.81	0.74	281.4
Word2Vec	0.85	0.81	0.829	435.3
LSTM	0.71	0.77	0.738	549.8
CNN	0.85	0.80	0.824	151.1
LSTM-CNN	**0.88**	**0.88**	**0.898**	**276.6**

Furthermore, we have provided the computation time of each model for both datasets to analyze the performance with time efficiency, as presented in Tables 3 and 4. The runtime of integrated LSTM-CNN for Twitter data is 449.3s. It is more

time-consuming as compared to CNN but more effective than the LSTM time. Meanwhile, apart from this execution time, the F-score is 17% and 8% better than the LSTM and CNN, respectively. In contrast, other models performed faster than the LSTM-CNN but with insignificant F-scores. In Table 4, the model computation time for the IMDB dataset is in the same sequence of Table 3. However, all model's performance time is more effective than Table 4. There can be two possible reasons for this effectiveness: (i) IMDB data volume is less, model's runtime is more petite, (ii) IMDB data is highly systemized.

The model's analysis on two different text data can be summarized as a neural network's efficiency for real-time sentiment classification is more effective than the ML models. The neural network methods can quickly analyze the growth of daily text data on various platforms. Simultaneously, the proposed WRS method with word embedding and integration of LSTM and CNN yielded improved accuracy for the real-time sentiment.

D. Comparative Analysis

We have compared our WRS method's performance with recently published work based on the sentiment classification's embedding and neural network methods. Table 5 provided the details of those work which have been compared with our work. It is evident that our method outperformed the given studies to a different extent. For instance, a recent study [9] also integrated LSTM-CNN for Wikipedia data and found a 65.9% F-score. However, our proposed work (WRS) has gained 88.1% and 89.8% F-score for Twitter and IMDB datasets, respectively.

TABLE V. Comparative Analysis Of Our Proposed Method's Performance With The Latest Existing Studies.

Authors	Models	Dataset	Method	F-score
Z. Rahimi [20]	Word2Vec	IMDB	Document-level sentiment	77.11
Q. Umer [19]	CNN	Platform Tweets	Priority Prediction	57.67
F. Huang [10]	LSTM	IMDB	integration with CNN	87.8
J.-L. Wu [9]	LSTM-CNN	Wikipedia	micro_f method with Word2vec	65.9
WRS (Proposed)	**LSTM-CNN**	**Twitter, IMDB**	**W2WG embedding**	**88.1, 89.8**

V. CONCLUSION

Recently, AI achieved a tremendous boost in real-time computing for text classification with NLP. Traditionally ML techniques provided limited accuracies with lexical dictionaries for real-time sentiment analysis. In this paper, we took advantage of neural networks by integrating the LSTM and CNN for the real-time sentiment. We have proposed a novel word-to-word graph (W2WG) embedding method as a pre-execution step for our proposed method, WRS. This method utilized two different datasets, our crawled Twitter dataset and IMDB dataset, for the experiments. The performance evaluation of experiments with different ML and neural network models shows that the WRS method with word embedding and integrated LSTM-CNN outperformed the other techniques and recently available studies for the real-time sentiment classification.

In the future, this study will be extended with different real-time datasets with other neural networks to experiment on the context-based sentiment.

REFERENCES

[1] M. Atzeni and D. Reforgiato Recupero, "Multi-domain sentiment analysis with mimicked and polarized word embeddings for human–robot interaction," Future Generation Computer Systems, vol. 110, pp. 984–999, Sep. 2020, doi: 10.1016/j.future.2019.10.012.

[2] B. Liu, "Preface," in Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. Cambridge: Cambridge University Press, pp. xi-xiv. 2015, doi: 10.1017/CBO9781139084789.001

[3] M. S. Neethu and R. Rajasree, "Sentiment analysis in twitter using machine learning techniques," presented at the 2013 Fourth International Conference on Computing, Communications and Networking Technologies, 2013, doi: 10.1109/icccnt.2013.6726818.

[4] S. Sagnika, B. S. P. Mishra, and S. K. Meher, "Improved method of word embedding for efficient analysis of human sentiments," Multimed Tools Appl, vol. 79, no. 43–44, pp. 32389–32413, Aug. 2020, doi: 10.1007/s11042-020-09632-9.

[5] D. Jatnika, M. A. Bijaksana, and A. A. Suryani, "Word2Vec Model Analysis for Semantic Similarities in English Words," Procedia Computer Science, vol. 157, pp. 160–167, 2019, doi: 10.1016/j.procs.2019.08.153.

[6] A. Rexha, M. Kröll, M. Dragoni, and R. Kern, "Polarity Classification for Target Phrases in Tweets: A Word2Vec Approach," in The Semantic Web, Springer International Publishing, 2016, pp. 217–223.

[7] X. Rong, "word2vec Parameter Learning Explained," in Arxiv. https://arxiv.org/pdf/1411.2738.pdf.

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

[9] J.-L. Wu, and K. R. Lai, "Identifying Emotion Labels From Psychiatric Social Texts Using a Bi-Directional LSTM-CNN Model," IEEE Access, vol. 8, pp. 66638–66646, 2020, doi: 10.1109/access.2020.2985228.

[10] F. Huang, and X. Li, "Attention-Emotion-Enhanced Convolutional LSTM for Sentiment Analysis," IEEE Trans. Neural Netw. Learning Syst., pp. 1–14, 2021, doi: 10.1109/tnnls.2021.3056664.

[11] J. Gehring, M. Auli, D. Grangier, "Convolutional sequence to sequence learning," in ArXiv, 2017, preprint arXiv:170503122.

[12] J. Zhou, S. Jin, and X. Huang, "ADeCNN: An Improved Model for Aspect-Level Sentiment Analysis Based on Deformable CNN and Attention," IEEE Access, vol. 8, pp. 132970–132979, 2020, doi: 10.1109/access.2020.3010802.

[13] A. L. Maas, T. Peter, "Learning word vectors for sentiment analysis," in Proc. 49th Annu. Meeting Assoc. Comput. Linguist. Human Lang. Technol., 2011, vol. 1, 2011, pp. 142–150.

[14] S. Baccianella, A. Esuli, "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining," In Proceeidngs of the LREC: 10, 2010, (pp. 2200–2204).

[15] S. M. Mohammad, "NRC-Canada: Building the state of-the-art in sentiment analysis of tweets," in arXiv, 2013, arXiv:1308.6242.

[16] E. Cambria, S. Poria, R. Bajpai, "SenticNet 4: a semantic resource for sentiment analysis based on conceptual primitives," In Proceedings of the COLING, 2016, (pp. 2666–2677).

[17] H. Cai, V. W. Zheng and K. C. Chang, "A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications," in IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 9, pp. 1616-1637, 1 Sept. 2018, doi: 10.1109/TKDE.2018.2807452.

[18] S. Ji, H. Yun, P. Yanardag, "Wordrank: Learning word embeddings via robust ranking," in ArXiv, 2015, preprint arXiv:1506.02761.

[19] Q. Umer, H. Liu, and I. Illahi, "CNN-Based Automatic Prioritization of Bug Reports," IEEE Trans. Rel., vol. 69, no. 4, pp. 1341–1354, Dec. 2020, doi: 10.1109/tr.2019.2959624.

[20] Z. Rahimi and M. M. Homayounpour, "TensSent: a tensor based sentimental word embedding method," Appl Intell, Jan. 2021, doi: 10.1007/s10489-020-02163-8.

[21] A. Rasool, and R. Tao, "GAWA–A Feature Selection Method for Hybrid Sentiment Classification," IEEE Access, vol. 8, pp. 191850–191861, 2020, doi: 10.1109/access.2020.3030642.

[22] A. Rasool, R. Tao, K. Marjan, and T. Naveed, "Twitter Sentiment Analysis: A Case Study for Apparel Brands," J. Phys.: Conf. Ser., vol. 1176, p. 022015, Mar. 2019, doi: 10.1088/1742-6596/1176/2/022015.