Received: 31 January 2020 | Revised: 7 April 2020 | Accepted: 20 May 2020

DOI: 10.1002/cpe.5909

WILEY

SPECIAL ISSUE PAPER

Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks

Aytuğ Onan

Department of Computer Engineering, Faculty of Engineering and Architecture, İzmir Katip Çelebi University, İzmir, Turkey

Correspondence
Aytuğ Onan, Department of Computer Engineering, Faculty of Engineering and Architecture, İzmir Katip Çelebi University, İzmir 35620, Turkey.
Email: aytug.onan@ikc.edu.tr

Funding information
İzmir Katip Çelebi University, Scientific Research Projects Coordination, Grant/Award Number: 2020-GAP-MÜMF-0006

Summary

Sentiment analysis is one of the major tasks of natural language processing, in which attitudes, thoughts, opinions, or judgments toward a particular subject has been extracted. Web is an unstructured and rich source of information containing many text documents with opinions and reviews. The recognition of sentiment can be helpful for individual decision makers, business organizations, and governments. In this article, we present a 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. The CNN-LSTM architecture consists of five layers, that is, weighted embedding layer, convolution layer (where, 1-g, 2-g, and 3-g convolutions have been employed), max-pooling layer, followed by LSTM, and dense layer. In the empirical analysis, the predictive performance of different word embedding schemes (ie, word2vec, fastText, GloVe, LDA2vec, and DOC2vec) with several weighting functions (ie, inverse document frequency, TF-IDF, and smoothed inverse document frequency function) have 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.

KEYWORDS

deep learning, LSTM, sentiment analysis, weighted word embeddings

1 | INTRODUCTION

Sentiment analysis (also known as, opinion mining) is one of the major tasks in natural language processing (NLP), in which attitudes, evaluations, thoughts, opinions, or judgments toward a particular subject has been extracted. With the developments in social media, the immense amount of user-generated unstructured text is available on the Web, containing thoughts and sentiments. The identification of sentiment can be essential for individual decision makers, business organizations, and governments.[1,2] Identifying public views with regard to policies, goods, and organizations can be of great benefit to organizations, decision support systems, and individuals.[3]

Methods of sentiment analysis can be generally divided into two groups, as lexicon-based and machine learning-based methods.[4] The lexicon-based methods identify the orientation of a text document by measuring the semantic orientation of words and sentences based on a dictionary.[5] By contrast, the machine learning-based methods utilize a labeled dataset, which serves as a training set to build classification models with the use of supervised learners, such as, Naïve Bayes, support vector machines, k-nearest neighbor algorithm, and random forest.[6,7]

Recent studies in the literature indicate that the processing of text documents with the use of deep neural networks in NLP tasks, such as, sentiment analysis, language modeling, and machine translation can significantly enhance the predictive performance.[8] Deep learning architectures, such as, recurrent neural networks (RNNs), convolutional neural networks (CNNs), gated recurrent units (GRUs), and long short-term memory (LSTM) architectures have been successfully employed for text mining applications.

The methods utilized in the representation of text documents are of great importance in the performance of machine learning-based NLP tasks.[9] The conventional text representation scheme is bag-of-words method, in which syntax, word rankings, and grammar rules of text are ignored. In addition, this representation scheme suffers from high dimensionality and sparsity of feature vectors.[10] Recent contributions in the field of NLP show that word embedding-based representation, in which text documents are represented in dense spaces through fixed-length vectors, significantly improves the performance.[11] Word embedding-based representation makes it possible to represent text in a more intense way with less dimension, thus eliminating the sparsity and high dimensionality problems encountered in bag-of-words scheme.[12] For the conventional word embedding schemes, such as, word2vec, equivalent weight values have been given to each word in the sentence and the final embedding has been obtained by computing the mean of word embeddings. Recent empirical analyzes in NLP suggest that the weighted word embeddings can improve performance in supervised and unsupervised NLP tasks.[13]

Considering the aforementioned issues, we present a deep learning-based architecture to sentiment analysis on product reviews obtained from Twitter, which combines TF-IDF weighted Glove word embedding with a novel CNN-LSTM-based architecture. The CNN-LSTM architecture consists of five layers, that is, embedding layer, convolution layer (where, 1-g, 2-g, and 3-g convolutions have been employed), max-pooling layer, followed by LSTM, and dense layer. In the empirical analysis, the predictive performance of different word embedding schemes (ie, word2vec, fastText, GloVe, LDA2vec, and DOC2vec) with several weighting functions [ie, inverse document frequency (IDF), TF-IDF, smoothed inverse document frequency (SIF)] have 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. To the best of our knowledge, the presented architecture is the first scheme for sentiment analysis in which weighted word embedding layer has been employed in conjunction with CNN-LSTM architecture. In addition, it is the first study, in which convolution layers with varying N-gram sizes have been utilized in conjunction with LSTM.

The structure of this article is a follows: In Section 2, we review the recent research contributions on deep learning-based sentiment analysis. In Section 3, we present the methodology of our research, including a description to the corpus, word embedding schemes, weighting functions, vector aggregation functions, conventional deep neural networks, and the proposed CNN-LSTM-based architecture. In Section 4, we present the experimental procedure and empirical results. Finally, Section 5 presents the concluding remarks.

2 | RELATED WORK

In the field of NLP, there are many studies using neural language models and deep learning architectures. This section briefly discusses the earlier research contributions on deep learning-based sentiment analysis, with a special emphasizes on CNNs and weighted word embeddings.

The study by Collobert and Weston[14] is one of the main studies using CNNs in the field of text mining. In this study, CNNs have been employed for named entity recognition and part of speech tagging. Similarly, Collobert et al[8] utilized CNN architecture for NLP tasks, such as part of speech tagging, chunking, named entity recognition, and semantic role labeling. In the study carried out by dos Santos and Gatti,[15] each character in the text document was represented as a fixed-length character vector. Relevant character-based embedding was used to extract morphological patterns in words. In addition, features were obtained using word2vec word embedding. The data representation obtained was applied in sentiment analysis on Twitter messages using an architecture with convolution layer and maximum pooling. In another study conducted by Kim,[16] basic CNN-based architecture was used for sentence level sentiment analysis. In another study, Zhang et al[17] utilized 9-29 layered CNNs to extract basic character-level features from text documents. The presented architecture achieved high predictive performance on large datasets, while it has obtained lower performance on smaller datasets, compared with the conventional artificial neural networks. In the study conducted by Johnson and Zhang,[18] word-level CNN architecture was developed for text classification. In the study, the predictive performance of deep word-level CNNs and character-level neural networks in text classification was compared and the higher performance was obtained by neural network architectures based on word-level convolution. Recently, Çano and Morisio[19] examined the performance of different N-gram convolution filters in CNN-based architecture for several NLP tasks, including, sentiment analysis on song lyrics, product, and movie reviews and sentence level sentiment analysis.

De Boom et al[20] presented a low-dimensional representation for short text documents, such as Twitter messages, based on semantic word embeddings and frequency information. The presented scheme employed a weighting scheme and a learning mechanism based on median-based loss function. In the study carried out by Schmidt,[21] word vectors obtained as a result of word2vec word embedding scheme were weighted using IDF and SIF. The performance of weighted word embeddings were evaluated on two datasets on hotel reservation reviews and question-answer forum posts. In the empirical analysis on English texts, IDF-based weighting function outperformed the other schemes. In another study, Rezaeinia et al[22] presented an ensemble word embedding scheme for sentiment analysis based on word2vec, POS2Vec, Lexicon2Vec, and word-position2vec schemes. Similarly, Onan[23] presented a two-stage topic extraction scheme for data analysis based on ensemble word embedding schemes and clustering. Djaballah et al[24] examined the predictive performance of machine learning and deep learning-based methods for sentiment analysis on Twitter messages. In the empirical analysis, word2vec word embedding representations have been obtained in two different ways, by unweighted and weighted vector aggregation. The empirical results indicated that weighted vector aggregation yields higher predictive performance. Curiskis

et al[2] examined the predictive performance of topic modeling schemes and neural language models on social network analysis. In the presented scheme, word2vec and doc2vec models have been weighted by TF-IDF measure. The weighted word embedding schemes outperformed the unweighted schemes for topic extraction.

Recently, Onan[5] employed three conventional word embedding schemes (ie, word2vec, fastText, and Glove) in conjunction with CNN on product reviews obtained from Twitter. In this scheme, a corpus containing 50 000 product reviews have been evaluated on conventional text representation schemes. In this article, we extend on this work and we present a novel deep learning-based architecture, which combines CNN with LSTM. The presented architecture consists of weighted word embedding layer, convolutional layer, max-pooling layer, LSTM layer, and dense layer. To represent text, we introduce TF-IDF weighted GloVe word embedding with center-based aggregation. In addition, we extend the size of corpus to 86 000 product reviews.

3 | METHODOLOGY

This section presents the methodology of our study, namely, descriptions to the corpus, neural language models, weighting functions, vector aggregation functions, and deep neural networks have been presented.

3.1 | Twitter product reviews corpus

To collect the Twitter product review corpus, we have followed the methodology set out in References 2,25. We have considered a range of keywords relevant to technological products to construct the corpus. We present an extended version of a corpus, initially employed in Reference 5. We have gathered about 93 000 tweets, written in English, with various topics related to technological products. Twitter4J, an open source Java library for Twitter Streaming API, was utilized to collect data.[5] We have employed automatic filtering on Twitter messages to delete duplicated posts, retweets, unclear, obsolete, and redundant tweets. To annotate raw twitter messages, we have performed an annotation process in which each message has been assigned either to positive or negative category, to indicate the sentiment orientation. Four experts have an notated the raw messages. We have computed Fleiss's kappa (κ) metric. For the corpus, we achieved a κ of 0.81, which indicated a perfect agreement among the four annotators.[7,26] In this way, we obtained a collection of roughly 50.000 positive tweets and roughly 43.000 negative tweets. Our final corpus contains a collection of tweets with 43.000 positive and 43.000 negative messages.[5] Twitter is composed of unstructured and irregular messages. Preprocessing is, therefore, an essential task for the analysis of sentiments on Twitter. To preprocess our corpus, we have adopted the framework set out in Reference 5. To examine the predictive performance of the proposed scheme, we have also utilized target-dependent Twitter corpus.[27] In this corpus, there are three class labels, as negative, neutral, and positive. The corpus consists of 6248 tweets in the training set and 692 tweets in the testing set.[27]

3.2 | Neural language models

Neural language models provide dense representation of text documents with semantic properties with less manual preprocessing. Dense vector representation yields promising results in NLP tasks. In this article, we have considered five word embedding schemes (ie, word2vec, fastText, Glove, LDA2vec, and Doc2vec). The rest of this section briefly presents the neural language models utilized in the empirical analysis:

- The word2vec model is an artificial neural network-based word embedding method consisting of the input layer, the output layer and the hidden layer.[9] It aims to learn the embedding of words by determining the probability that a particular word will be surrounded by other words. The model has two basic architectures, skip-gram (SG) and continuous bag-of-words (CBOW). The CBOW architecture defines the target word by taking the content of each word as input; SG architecture, on the other hand, predicts the words surrounding the target word by taking the target word as input.[7] The CBOW architecture can work properly with a small amount of data. The SG architecture works better on large datasets.
- The GloVe model is a word2vec-based model developed to effectively learn word embeddings from text documents. The model combines local context-based learning of the word2vec model with global matrix factorization.[7,28] In the model, probability ratios of words are also taken into account in calculating the error function. Words that are observed closely in the text document and are likely to be seen together are more important than other words in the learning process.[28]
- The fastText model is another effective method to obtain word embeddings from text documents. In this model, each word is represented by dividing the character into n-grams. Word vectors are created for each n-gram in the training set. The fastText model provided a more efficient word embedding scheme for morphologically rich languages and rare words[7,29]
- The LDA2vec model[7,30] is another word embedding scheme based on the word2vec model and the latent Dirichlet allocation algorithm. In this method, dense word vectors from document-level mixture vectors are obtained depending on the Dirichlet distribution. The model allows to

identify the topics of text collections and word vectors are adjusted according to the subject. In this context, the model connects each word to a subject and a word embedding scheme enriched with the subject has been identified.

- The Doc2vec method is an unsupervised neural language model based on word2vec to obtained vector-based representation for sentences, paragraphs, and documents.[9] Unlike the word2vec model, the Doc2vec model provides a feature vector with an additional context for each document contained in the text corpus. This document vector is trained along with word vectors.

3.3 | Weighting functions

In the neural language models, such as word2vec, the equivalent weights were allocated to each word in the sentence, and the word embedding was obtained by taking the mean of embedding terms. Recent empirical analysis of NLP tasks shows that weighted averages of word embedding can improve performance in clustering or classification.[31] We have considered three weighting functions in the empirical analysis (ie, TF-IDF, IDF, and SIF). The remainder of this section outlines briefly the weighting functions used in empirical analysis:

IDF is one of the most commonly utilized weighting schemes in information retrieval. Let D denote the total number of documents in text corpus and D_i denote the total number of documents in which word i has been encountered. IDF has been computed based on Equation (1)[31]:

$$\mathrm{idf}(i) = \left(\frac{D}{D_i}\right). \tag{1}$$

SIF is an improved weighting function to assign weight values to terms in text documents.[30] Let n_i denote the total number of times word i encountered in the document and N denote the total number of terms encountered in text corpus. Term frequency (tf_i) for a particular word i has been computed based on Equation (2).

$$\mathrm{tf}_i = \left(\frac{n_i}{N}\right). \tag{2}$$

TF-IDF is another common weighting scheme based on term frequency and IDF. Term frequency represents the relative frequency of a word i in a text document and IDF scales with the number of documents in which the word has been encountered. Based on term frequency (tf_i) (as computed by Equation (2)) and IDF (idf(i)) (as computed by Equation (1)), TF-IDF has been computed based on Equation (3):

$$\mathrm{TF} - \mathrm{IDF} = \mathrm{tf}_i * \mathrm{idf}(i). \tag{3}$$

Taking term frequencies into account, SIF has been computed based on Equation (4), where a is a parameter, which has been generally set to 10^{-4} and tf_{ic} denote the corpus wide term frequencies[31]:

$$w_i = \left(\frac{a}{a + \mathrm{tf}_{ic}}\right). \tag{4}$$

3.4 | Vector aggregation functions

In this study, three basic vector aggregation methods (ie, weighted sum, center-based aggregation, and Delta rule) have been taken into consideration.

The weighted sum scheme is one of the basic method utilized in vector aggregation. In this method, the weighted averages of the vectors in a given text document are calculated according to Equation (5)[21]:

$$c = \sum_i w_i \mathrm{tf}_i v_i. \tag{5}$$

Here, tf_i represents the incidence of the word i in the text, and v_i represents the vector for the word i.

The second method considered in vector aggregation is center-based aggregation. In this method, unlike the weighted sum method, a point representing the center of the text documents has been computed once and it has been reused to effectively embed multiple documents. The center-based aggregation method is computed according to Equation (6)[21]:

$$c = \sum_{i \in V_d} w_i \mathrm{tf}_i v_i - \sum_{i \in V} w_i \mathrm{tf}_{ic} v_i. \tag{6}$$

In the center-based aggregation method, the word embedding representation for each word is determined based on the center point of the document. Unlike the center-based aggregation method, in aggregation according to the Delta rule, rare words are ignored in the process of obtaining a neural language representation. Delta rule aggregation is calculated according to Equation (7)[21]:

$$c = \sum_{i \in V_d} w_i \mathrm{tf}_i v_i - \sum_{i \in V_d} w_i \mathrm{tf}_{ic} v_i. \tag{7}$$

3.5 | Deep neural network architectures

In a wide range of applications, deep learning architectures have already been used, including computer vision, pattern recognition, and NLP. Deep learning architectures enable to learn multi-level feature representations. The architectures seek to identify learning models based on multiple layers of nonlinear information processing in a hierarchical way.[32] Deep learning architectures, such as, RNNs, CNNs, GRUs, and LSTM architectures have been successfully employed for text mining applications.

CNNs are deep neural network-based architectures that process data using a topology based on a grid. A special type of mathematical operation, called convolution, has been employed on CNN. The convolution operation has been employed in one or more convolution layers. Typical CNN architecture consists of input layer, output layer, and hidden layers. The hidden layers of CNN consist of several layers, namely, convolutional layers, pooling layers, and fully connected layers. In convolutional layers, feature maps have been obtained from the input data by means of convolution operation. The activation functions (such as, reLU) has been employed in conjunction with feature maps so that nonlinearity can be added to the architecture.[7] In pooling layers, the outputs of neuron clusters have been combined. It reduced the spatial scale of function spaces, and improved the capacity of the models to accommodate overfitting. In pooling layer, maximum pooling has been employed. The fully connected layers have been utilized to obtain the final output of the architecture.[33]

RNN is a deep network architecture used to process sequential data. In this architecture, connections between neurons form a directed graph.[7] The RNN architecture makes it possible to process inputs of any length. During the processing of data, historical information is also taken into account. RNN architectures suffer from vanishing gradient problem. Long-term dependencies are difficult to model in RNN architectures due to the exploding or vanishing gradients.[34]

LSTMs is another deep neural network architecture based on RNNs. In conventional RNN, arbitrarily long sequences of input cannot be properly handled. To deal with this problem, LSTM contains forget gates. In this architecture, the backpropagation of error has been allowed until a limited number of time steps. For a typical LSTM unit, there are three kinds of gates: input gate, output gate, and forget gate. The status of the gates control which information should be kept and when the information should be accessed.[35]

GRU is another deep learning architecture based on RNNs. In a typical GRU architecture, there are two gates (ie, reset gate and update gate).[36]

3.6 | Proposed CNN-LSTM architecture

The presented CNN-LSTM architecture combines TF-IDF weighted GloVe word embedding scheme with CNN-LSTM architecture. The general structure of CNN-LSTM architecture has been presented in Figure 1. The CNN-LSTM architecture consists of several layers, that is, weighted embedding layer, dropout, convolution layer, maximum pooling layer, LSTM, and dense layer. In the weighted embedding layer, we have made empirical analysis with different word embedding schemes (ie, word2vec, fastText, GloVe, LDA2vec, and DOC2vec) with several weighting functions (ie, IDF, TF-IDF, and SIF). Since TF-IDF weighted GloVe word embedding with center-based aggregation yields the highest predictive performance, the final architecture utilizes this scheme in the weighted word embedding layer. In the embedding layer of CNN architecture, text documents are represented using word embedding methods. Since pretrained word vectors yield higher predictive performance, pretrained word embedding scheme have been utilized in this layer. Based on the empirical results, the vector size of 300 is taken for word embedding. In the architecture, dropout layer has followed weighted word embedding layer in order to prevent overfitting. After dropout layer, convolution layer has been configured. In this layer, using the stack of convolution layers (1-g, 2-g, and 3-g), extraction of 1-g, 2-g, and 3-g based attributes is performed. A fixed number of 80 filters are applied on each layer to create feature maps. Initially, any input text was converted into a concatenation of all word vectors. Those vectors are word embedding schemes, which capture syntactic and semantic meanings of words. The input text messages have been represented as a sequence of vectors as $V = [v_1, v_2, \ldots, v_n]$. The vector sequence was transformed into a matrix M based on the word embedding dimension and the length of the input text. Then, convolutional layer was utilized to derive the local features. To extract local features, convolutional layer slides continuously window-shaped filters with rows of M. The width of filters has been taken as the same as the width of the word embedding. The filters slide over M and perform convolution operations. Let $M[i{:}\ j]$ denote a submatrix of M from row i to row j, the output of the convolutional layer for ith filter is calculated according to Equations (8) and (9):

$$o_i = M[i : i + h - 1] \otimes w_i, \tag{8}$$

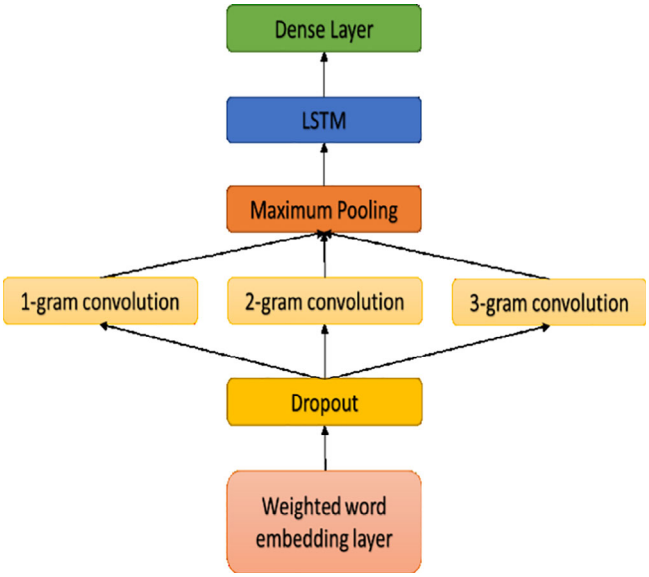

FIGURE 1 The general structure of CNN-LSTM architecture. CNN-LSTM, convolutional neural network-long short-term memory network

$$c_i = f(o_i + b), \tag{9}$$

where w_i denotes the ith filter, $\otimes$ denotes element-wise multiplication, b denotes bias, f denotes the activation function, h denotes the height of the filter, and c_i denotes the feature extracted by the corresponding filter. Due to its high experimental performance, ReLU activation function is utilized in the layers as an activation function. After the convolution layer, maximum pooling has been employed. The CNN-LSTM architecture combines two previously employed deep neural network models together. The convolution and maximum pooling layers have been followed by LSTM. The dense layer is a feed-forward neural network where the feature maps obtained in the previous layers are given as input. The dense layer contains 80 neural network cells. In the model, L2 norm (with 0.1 regularization coefficient) and dropout (with 0.30 coefficient) were applied to prevent overfitting. To calculate the loss value and optimize the training process, binary cross entropy, and the Adam algorithm are used. In the experimental analysis, the dataset was partitioned as 70/10/20 and used during the training, development, and testing phases, respectively.

4 | EXPERIMENTS AND RESULTS

This section presents the evaluation measures, the experimental procedure, and the empirical results of the study.

4.1 | Experimental procedure

In the empirical analysis, classification accuracy has been utilized as the evaluation measure. In the empirical analysis, the predictive performance of different word embedding schemes (ie, word2vec, fastText, GloVe, LDA2vec, and DOC2vec) with several weighting functions (ie, IDF, TF-IDF, SIF) have been evaluated in conjunction with conventional deep neural network architectures. We used Tensorflow and Keras to implement and train the architectures that are based on deep learning used in empirical analysis. We used hyperparameter search algorithms for each model to get maximum predictive efficiency from each deep learning model. For this, optimization of hyper parameters based on Bayesian optimization using Gaussian method was used. For word2vec and fastText schemes, we have employed continuous SG and CBOW schemes, with vector sizes of 200 and 300. For LDA2vec, the parameter set has been set out based on Reference 7. For word2vec and fastText models, SG scheme yields higher predictive performance with a vector size of 300. Hence, the empirical results for these configurations have been listed in Section 4.2. The results listed in Section 4.2 report the predictive performances on test sets.

4.2 | Experimental results

In Table 1, the experimental results obtained by conventional neural language models and deep learning architectures have been presented. For the word embedding schemes listed in Table 1, we consider unweighted models, namely, the equivalent weights were allocated to each word in

TABLE 1 Experimental results obtained by conventional word embedding schemes and architectures

Twitter product reviews corpus					
Word embedding scheme	**CNN**	**RNN**	**LSTM**	**GRU**	**CNN-LSTM**
word2vec	68,62	72,27	73,79	72,25	77,61
GloVe	72,34	76,01	77,03	75,93	80,92
fastText	72,26	74,53	76,78	75,79	79,67
LDA2vec	70,16	73,78	76,71	75,57	79,03
Doc2vec	69,94	72,64	76,61	74,90	79,00
word2vec + sentence padding	75,28	78,72	79,89	79,72	83,52
GloVe + sentence padding	79,55	83,08	84,08	83,71	87,65
fastText + sentence padding	79,46	81,95	83,86	83,71	86,68
LDA2vec + sentence padding	78,55	81,37	83,84	82,09	86,23
Doc2vec + sentence padding	78,07	80,61	83,46	81,75	86,18
Weighted word embedding	87,12	87,60	91,11	89,00	93,85
Target-dependent Twitter corpus					
Word embedding scheme	CNN	RNN	LSTM	GRU	CNN-LSTM
word2vec	61,73	65,38	66,90	65,35	70,72
GloVe	65,45	69,12	70,14	69,03	74,03
fastText	65,37	67,64	69,89	68,90	72,77
LDA2vec	63,27	66,89	69,82	68,67	72,13
Doc2vec	63,05	65,74	69,72	68,00	72,10
word2vec + sentence padding	70,43	73,40	74,65	73,97	78,77
GloVe + sentence padding	74,21	77,41	78,95	78,23	84,22
fastText + sentence padding	74,15	76,30	78,43	77,88	82,96
LDA2vec + sentence padding	72,06	76,09	78,16	77,75	80,83
Doc2vec + sentence padding	72,01	75,02	78,10	77,57	80,82
Weighted word embedding	80,22	80,71	84,22	82,11	86,93

Abbreviations: CNN, convolutional neural network; RNN, recurrent neural network; LSTM, long short-term memory; GRU, gated recurrent unit; CNN-LSTM, proposed architecture, weighted word embedding; TF-IDF, weighted GloVe word embedding with center-based aggregation.

the sentence, and the word embedding was obtained by taking the mean of embedding terms. To examine the impact of the weighted aggregation on the predictive performance, we have also included empirical results with sentence padding and the weighted word embedding scheme (TF-IDF weighted Glove with center-based aggregation) in Table 1.

Regarding the predictive performance of conventional deep learning models presented in Table 1, the highest predictive performance has been achieved by LSTM architecture, which is followed by GRUs. The third highest predictive performance among the conventional deep neural networks has been achieved by RNN. The lowest predictive performances have been obtained by CNN architecture. For the unweighted neural language models, the proposed architecture, which combines CNN and LSTM, outperforms the other compared architectures. Regarding the predictive performance of unweighted neural language models, the highest predictive performance has been achieved by GloVe. The second highest predictive performance has been achieved by fastText. LDA2vec has obtained the third highest predictive performance and Doc2vec has achieved the fourth highest predictive performance. As it can be seen from the accuracy values listed in Table 1, the lowest predictive performances have been generally obtained by word2vec model. As it can be observed from the empirical results listed in Table 1, sentence padding yields higher predictive performances compared with the unweighted word embedding schemes. However, the weighted word embedding scheme (TF-IDF weighted Glove with center-based aggregation) outperforms the padding-based configurations.

In Table 2, the predictive performance of different word embedding schemes, weighting functions, vector aggregation functions, and deep learning architectures on Twitter product reviews corpus have been presented. To summarize the main empirical results, Figure 2 also presents the

TABLE 2 Experimental results obtained by weighted word embedding schemes on Twitter product reviews corpus

Word embedding scheme	Weighting function	Vector aggregation	CNN	RNN	LSTM	GRU	Proposed architecture
Doc2vec	IDF-weighted	Weighted sum	77,68	80,60	84,22	82,98	85,74
Doc2vec	SIF-weighted	Weighted sum	78,10	80,76	84,29	83,06	85,99
Doc2vec	TF-IDF-weighted	Weighted sum	78,15	80,83	84,54	83,19	86,18
Doc2vec	IDF-weighted	Center-based aggregation	78,93	82,08	84,97	83,33	86,44
Doc2vec	SIF-weighted	Center-based aggregation	79,03	82,21	85,09	83,72	86,48
Doc2vec	TF-IDF-weighted	Center-based aggregation	79,62	82,31	85,10	83,92	86,60
Doc2vec	IDF-weighted	Delta rule	78,39	81,41	84,63	83,23	86,22
Doc2vec	SIF-weighted	Delta rule	78,73	82,03	84,89	83,24	86,40
Doc2vec	TF-IDF-weighted	Delta rule	78,77	82,08	84,96	83,32	86,42
fastText	IDF-weighted	Weighted sum	79,72	82,32	85,12	84,60	86,61
fastText	SIF-weighted	Weighted sum	79,77	82,36	85,22	84,77	86,69
fastText	TF-IDF-weighted	Weighted sum	80,13	82,50	86,08	84,91	86,79
fastText	IDF-weighted	Center-based aggregation	80,94	84,31	87,06	85,47	87,99
fastText	SIF-weighted	Center-based aggregation	81,27	84,51	87,18	85,62	88,01
fastText	TF-IDF-weighted	Center-based aggregation	81,41	84,78	87,25	85,96	88,26
fastText	IDF-weighted	Delta rule	80,32	82,98	86,28	85,07	86,94
fastText	SIF-weighted	Delta rule	80,45	83,31	86,48	85,18	86,95
fastText	TF-IDF-weighted	Delta rule	80,85	84,17	86,92	85,34	86,99
GloVe	IDF-weighted	Weighted sum	81,53	85,34	87,30	86,19	88,54
GloVe	SIF-weighted	Weighted sum	82,18	85,46	87,63	86,22	89,01
GloVe	TF-IDF-weighted	Weighted sum	82,36	85,69	88,24	86,72	89,51
GloVe	IDF-weighted	Center-based aggregation	84,81	87,28	89,78	87,66	91,19
GloVe	SIF-weighted	Center-based aggregation	85,37	87,57	90,31	88,88	92,24
GloVe	TF-IDF-weighted	Center-based aggregation	87,12	87,60	91,11	89,00	93,85
GloVe	IDF-weighted	Delta rule	82,57	85,79	88,65	86,78	90,03
GloVe	SIF-weighted	Delta rule	82,58	86,10	89,08	86,96	90,75
GloVe	TF-IDF-weighted	Delta rule	83,02	86,76	89,67	87,08	90,84
LDA2vec	IDF-weighted	Weighted sum	75,83	79,52	81,16	81,08	84,52
LDA2vec	SIF-weighted	Weighted sum	75,92	79,54	81,98	81,26	84,73
LDA2vec	TF-IDF-weighted	Weighted sum	76,02	80,02	82,89	81,36	84,82
LDA2vec	IDF-weighted	Center-based aggregation	77,21	80,26	83,73	82,49	85,33
LDA2vec	SIF-weighted	Center-based aggregation	77,22	80,42	83,84	82,69	85,63
LDA2vec	TF-IDF-weighted	Center-based aggregation	77,46	80,60	84,10	82,94	85,73
LDA2vec	IDF-weighted	Delta rule	76,14	80,10	83,18	81,53	84,96
LDA2vec	SIF-weighted	Delta rule	76,80	80,10	83,53	82,24	85,05
LDA2vec	TF-IDF-weighted	Delta rule	76,95	80,21	83,67	82,32	85,17
word2vec	IDF-weighted	Weighted sum	74,61	78,44	79,80	79,24	83,10
word2vec	SIF-weighted	Weighted sum	74,87	78,53	79,92	79,41	83,31
word2vec	TF-IDF-weighted	Weighted sum	74,95	78,55	80,17	79,46	83,52
word2vec	IDF-weighted	Center-based aggregation	74,96	78,70	80,21	80,05	83,59
word2vec	SIF-weighted	Center-based aggregation	75,10	78,87	80,38	80,39	84,05
word2vec	TF-IDF-weighted	Center-based aggregation	75,39	78,89	80,57	80,41	84,05
word2vec	IDF-weighted	Delta rule	75,51	78,90	80,72	80,60	84,06
word2vec	SIF-weighted	Delta rule	75,60	78,91	80,94	80,71	84,06
word2vec	TF-IDF-weighted	Delta rule	75,75	78,91	80,95	80,76	84,09

Abbreviations: CNN, convolutional neural network; RNN, recurrent neural network; LSTM, long short-term memory; GRU, gated recurrent unit; CNN-LSTM, proposed architecture, weighted word embedding; TF-IDF, weighted GloVe word embedding with center-based aggregation.

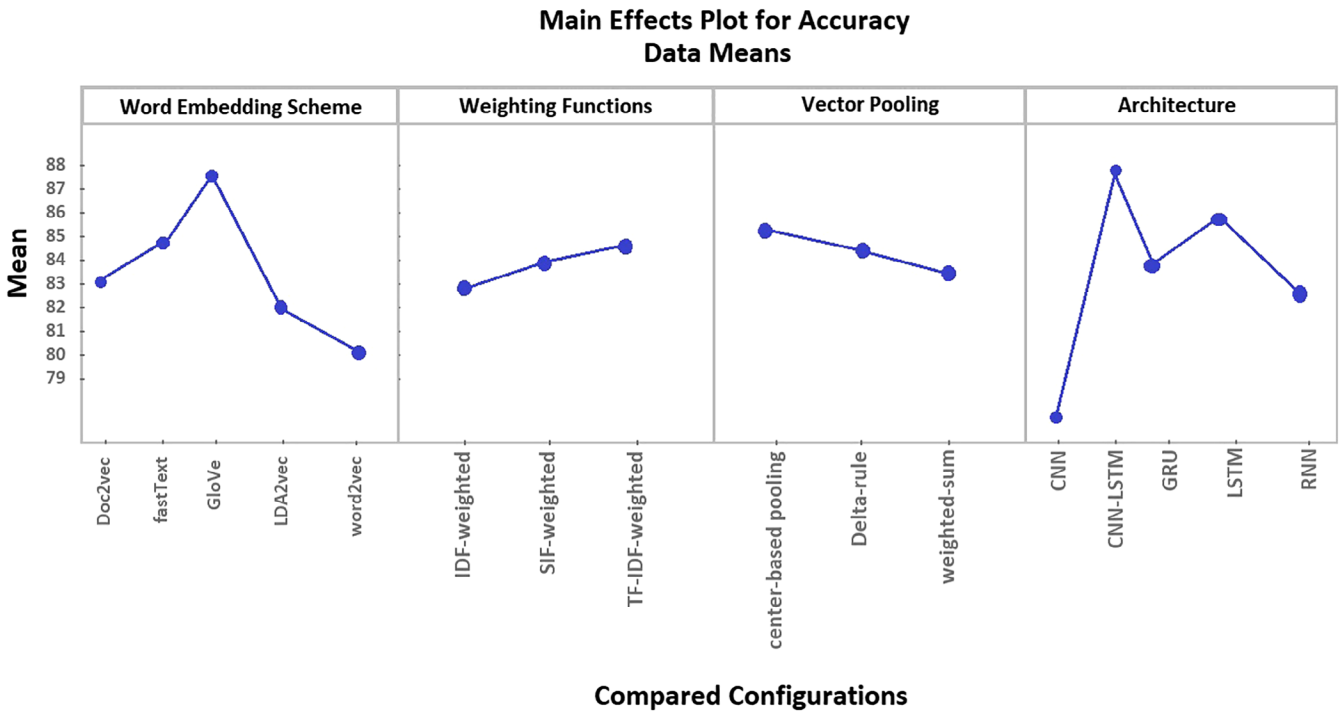

FIGURE 2 Main effects plot for compared configurations

main effects plot for the analyzed factors. As it can be observed from the results presented in Table 2, TF-IDF weighted word embedding schemes outperform the two other weighting functions (ie, IDF-weighing and SIF-weighting). Regarding the performance of vector aggregation functions, center-based aggregation function outperforms the Delta rule and weighted sum. The second highest predictive performances have been obtained by the Delta rule aggregation function and the lowest predictive performances have been obtained by the weighted sum. Regarding the predictive performance of deep neural network architectures, the proposed architecture presented in Figure 1 outperforms the conventional deep neural networks. Regarding the predictive performance of word embedding schemes, GloVe scheme outperforms the other neural language modeling schemes. In addition, Table 3 presents the predictive performance of different word embedding schemes, weighting functions, and vector aggregation functions on target-dependent Twitter corpus. As it can be observed from the empirical results listed in Table 3, the same patterns are valid for target-dependent Twitter corpus.

Based on the empirical analysis on word embedding schemes, weighting functions, vector aggregation functions, and deep neural network architectures, several insights follow:

- The utilization of weighting functions (such as, IDF, SIF, and TF-IDF) in conjunction with word embedding schemes enhance the predictive performance of word embedding schemes in text representation. The utilization of weighted word embedding schemes yield higher predictive performance compared with the unweighted word embedding schemes. These empirical results are in line with empirical results on weighted document vector embedding on TripAdvisor reviews.[21]
- In the empirical analysis, five word embedding schemes (ie, Doc2vec, fastText, GloVe, LDA2vec, and word2vec) have been considered. The experimental results indicate that GloVe scheme outperforms the other word embedding schemes. These empirical results are in line with empirical results on several NLP tasks, such as, word anology, named entity recognition, word similarity tasks. GloVe yields higher predictive performance compared with word2vec on the same vocabulary, corpus, training time, and window size.[28] This is because GloVe contains even statistical information on the words in the corpus.
- In the empirical analysis, the predictive performance of three different weighting functions has been considered. The empirical results indicate that TF-IDF weighted word embedding yields higher predictive performance compared with IDF-weighted word embedding and SIF-weighted word embedding.
- In the empirical analysis, the predictive performance of three different vector aggregation functions (namely, center-based aggregation, Delta rule, and weighted sum) has been considered. The experimental results indicate that center-based aggregation yields higher predictive performance compared with the other schemes.
- To examine the impact of weighted aggregation, we have also analyzed the predictive performance on unweighted word embedding schemes with sentence padding. The empirical results indicate that the utilization of word embedding schemes in conjunction with sentence padding can

TABLE 3 Experimental results obtained by weighted word embedding schemes on target-dependent Twitter corpus

Word embedding scheme	Weighting function	Vector aggregation	CNN	RNN	LSTM	GRU	Proposed architecture
Doc2vec	IDF-weighted	Weighted sum	70,78	73,71	77,33	76,08	78,82
Doc2vec	SIF-weighted	Weighted sum	71,21	73,86	77,39	76,16	79,07
Doc2vec	TF-IDF-weighted	Weighted sum	71,26	73,94	77,65	76,30	79,26
Doc2vec	IDF-weighted	Center-based aggregation	72,04	75,19	78,08	76,43	79,52
Doc2vec	SIF-weighted	Center-based aggregation	72,13	75,31	78,19	76,83	79,57
Doc2vec	TF-IDF-weighted	Center-based aggregation	72,72	75,42	78,21	77,02	79,68
Doc2vec	IDF-weighted	Delta rule	71,49	74,51	77,73	76,33	79,30
Doc2vec	SIF-weighted	Delta rule	71,83	75,14	78,10	76,35	79,48
Doc2vec	TF-IDF-weighted	Delta rule	71,88	75,19	78,06	76,42	79,50
fastText	IDF-weighted	Weighted sum	72,82	75,42	78,23	77,71	79,69
fastText	SIF-weighted	Weighted sum	72,88	75,47	78,32	77,88	79,77
fastText	TF-IDF-weighted	Weighted sum	73,24	75,61	79,19	78,02	79,87
fastText	IDF-weighted	Center-based aggregation	74,05	77,42	80,17	78,57	81,07
fastText	SIF-weighted	Center-based aggregation	74,38	77,62	80,29	78,72	81,09
fastText	TF-IDF-weighted	Center-based aggregation	74,52	77,89	80,36	79,07	81,34
fastText	IDF-weighted	Delta rule	73,43	76,09	79,39	78,18	80,02
fastText	SIF-weighted	Delta rule	73,56	76,41	79,59	78,29	80,03
fastText	TF-IDF-weighted	Delta rule	73,95	77,27	80,03	78,45	80,07
GloVe	IDF-weighted	Weighted sum	74,63	78,45	80,40	79,29	81,62
GloVe	SIF-weighted	Weighted sum	75,28	78,57	80,73	79,33	82,09
GloVe	TF-IDF-weighted	Weighted sum	75,47	78,80	81,35	79,82	82,59
GloVe	IDF-weighted	Center-based aggregation	77,92	80,38	82,89	80,77	84,27
GloVe	SIF-weighted	Center-based aggregation	78,48	80,68	83,42	81,98	85,33
GloVe	TF-IDF-weighted	Center-based aggregation	80,22	80,71	84,22	82,11	86,93
GloVe	IDF-weighted	Delta rule	75,68	78,89	81,76	79,89	83,12
GloVe	SIF-weighted	Delta rule	75,68	79,21	82,19	80,07	83,83
GloVe	TF-IDF-weighted	Delta rule	76,12	79,86	82,78	80,19	83,92
LDA2vec	IDF-weighted	Weighted sum	68,94	72,63	74,27	74,18	77,60
LDA2vec	SIF-weighted	Weighted sum	69,03	72,65	75,09	74,36	77,81
LDA2vec	TF-IDF-weighted	Weighted sum	69,12	73,13	75,99	74,47	77,90
LDA2vec	IDF-weighted	Center-based aggregation	70,31	73,37	76,84	75,60	78,41
LDA2vec	SIF-weighted	Center-based aggregation	70,33	73,52	76,95	75,80	78,72
LDA2vec	TF-IDF-weighted	Center-based aggregation	70,57	73,71	77,21	76,05	78,82
LDA2vec	IDF-weighted	Delta rule	69,25	73,20	76,29	74,64	78,04
LDA2vec	SIF-weighted	Delta rule	69,91	73,20	76,63	75,35	78,13
LDA2vec	TF-IDF-weighted	Delta rule	70,05	73,32	76,77	75,43	78,26
word2vec	IDF-weighted	Weighted sum	67,72	71,55	72,90	72,35	76,18
word2vec	SIF-weighted	Weighted sum	67,97	71,64	73,03	72,52	76,39
word2vec	TF-IDF-weighted	Weighted sum	68,06	71,66	73,27	72,57	76,60
word2vec	IDF-weighted	Center-based aggregation	68,07	71,81	73,32	73,15	76,67
word2vec	SIF-weighted	Center-based aggregation	68,21	71,98	73,49	73,50	77,13
word2vec	TF-IDF-weighted	Center-based aggregation	68,50	71,99	73,67	73,51	77,13
word2vec	IDF-weighted	Delta rule	68,62	72,01	73,83	73,71	77,14
word2vec	SIF-weighted	Delta rule	68,71	72,01	74,05	73,82	77,14
word2vec	TF-IDF-weighted	Delta rule	68,86	72,02	74,06	73,87	77,17

Abbreviations: CNN, convolutional neural network; RNN, recurrent neural network; LSTM, long short-term memory; GRU, gated recurrent unit; CNN-LSTM, proposed architecture, weighted word embedding; TF-IDF, weighted GloVe word embedding with center-based aggregation.

yield higher performance compared with the utilization of unweighted word embedding schemes (such as, word2vec, fastText, and Glove). Yet, the weighted word embedding schemes outperform the empirical results obtained with sentence padding.

- In the empirical analysis, we have considered five deep neural networks (ie, CNNs, RNNs, LSTM, GRU, and CNN-LSTM). The highest predictive performance has been obtained by CNN-LSTM architecture and the second highest predictive performance has been obtained by LSTM. LSTM has been shown to be an effective technique on NLP tasks, including sentiment analysis.[37] The hybrid deep neural architectures yield promising results on NLP tasks. The hybrid architecture based on CNN and LSTM outperforms the conventional deep neural networks. The experimental results are in line with recent empirical results on text classification.[38]
- In the proposed architecture, dropout has been utilized in order to prevent overfitting.[39] In addition, a holdout set has been utilized to examine the predictive performance and generalization ability of the architecture on unseen instances. The predictive performances reported on test model are promising.
- For text sentiment analysis, the document length and the characteristics of input data can be critical. In the empirical analysis, two corpus have been employed to validate the predictive performance of the proposed scheme. However, document length can be a constraint when adjusting the parameters of word embeddings and there is no single approach that may yield the highest predictive performance on all type of text classification tasks.

5 | CONCLUSION

This article has proposed an effective deep learning-based architecture for sentiment analysis. The presented architecture combines TF-IDF weighted Glove word embedding with CNN-LSTM architecture. The CNN-LSTM architecture consists of five layers, that is, weighted embedding layer, convolution layer (where, 1-g, 2-g, and 3-g convolutions have been employed), max-pooling layer, followed by LSTM, and dense layer. To verify the effectiveness of the presented scheme, the predictive performance of different word embedding schemes (ie, word2vec, fastText, GloVe, LDA2vec, and DOC2vec) with several weighting functions (ie, IDF, TF-IDF, and SIF function) have been evaluated in conjunction with conventional deep neural network architectures. The empirical results indicate that weighted word embedding schemes outperform unweighted word embedding schemes. In addition, the utilization of CNN in conjunction with LSTM architecture can yield promising results. Among the compared configurations, the highest predictive performance has been obtained by the proposed scheme with a classification accuracy of 93.85%.

ORCID

Aytuğ Onan https://orcid.org/0000-0002-9434-5880

REFERENCES

1. Onan A, Korukoğlu S, Bulut H. A hybrid ensemble pruning approach based on concensus clustering and multi-objective evolutionary algorithm for sentiment classification. *Inf Process Manag*. 2017;53(4):814-833.
2. Curiskis A, Drake B, Osborn TR, Kennedy PJ. An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit. *Information Processing and Management*. 2019;57(2):102034.
3. Fersini E, Messina E, Pozzi FA. Sentiment analysis: Bayesian ensemble learning. *Decis Support Syst*. 2014;68:26-38.
4. Medhat W, Hassan A, Korashy H. Sentiment analysis algorithms and applications: a survey. *Ain Shams Eng J*. 2014;5(4):1093-1113.
5. Onan A. Deep learning based sentiment analysis on product reviews on Twitter. Paper presented at: Proceedings of International Conference on Big Data Innovations and Applications; August ; 2019:80-91; Springer: Berlin.
6. Onan A. Sarcasm identification on twitter: a machine learning approach. Paper presented at: Proceedings of CSOC 2017; 2017:374-383; Springer, Berlin, Germany.
7. Onan A. Mining opinions from instructor evaluation reviews: a deep learning approach. *Comput Appl Eng Educ*. 2020;28(1):117-138.
8. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing almost from scratch. *J Mach Learn Res*. 2011;12:2493-2537.
9. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations. arXiv preprint 2013. https://arxiv.org/abs/1301.3781. January 30, 2020.
10. Hackeling G. *Mastering Machine Learning with Scikit-Learn*. New York, NY: Packt Publishing; 2017:1-25.
11. Güngör O, Üsküdarlı S, Güngör T. Recurrent neural network for Turkish named entity recognition. Paper presented at: Proceedings of the 26th Signal Processing and Communication Applications Conference; April 2018:1-4; IEEE, New York, NY.
12. Yıldırım S, Yıldız T. Türkçe için karşılaştırmalı metin sınıflandırma analizi. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*. 2018;24(5):879-886.
13. Gupta V, Saw AK, Talukdar PP, Netrapalli P. Unsupervised document representation using partition word-vectors averaging. Proceedings of the 7th International Conference on Learning Representations; May 2018:1-28; IEEE, New York, NY.
14. Collobert R, Weston J. A unified architecture for natural language processing: deep neural network with multitask learning. Paper presented at: Proceedings of the 25th International Conference on Machine Learning; May 2008:160-167; IEEE, New York, NY.
15. Dos Santos, C., Gatti, M. Deep convolutional neural networks for sentiment analysis for short texts. Paper presented at: Proceedings of the 25th International Conference on Computational Linguistics; August 2014:69-78; ACL, New York, NY.
16. Kim Y. Convolutional neural networks for sentence classification 2014. arXiv preprint. https://doi.org/10.3115/v1/D14-1181

17. Zhang X, Zhao J, LeCun, Y. Character-level convolutional networks for text classification. Proceedings of the 28th NIPS; December 2015:649-657; IEEE, New York, NY.
18. Johnson R, Zhang T. Deep pyramid convolutional neural networks for text categorization. Paper presented at: Proceedings of ACL 2017. August 2017:562-570; ACL, New York, NY.
19. Çano E, Morisio M. Role of data properties on sentiment analysis of texts via convolutions. Paper presented at: Proceedings of WorldCIST 2018; April 2018:330-337.
20. De Boom C, Van Canneyt S, Demeester T, Dhoedt B. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recogn Lett*. 2016;80(1):150-156.
21. Schmidt CW. Improving a tf-idf weighted document vector embedding; 2019; arXiv preprint. https://arxiv.org/abs/1902.09875. January 30, 2020.
22. Rezaeinia SM, Rahmani R, Ghodsi A, Veisi H. Sentiment analysis based on improved pretrained word embeddings. *Expert Syst Appl*. 2019;117:139-147.
23. Onan A. Two-stage topic extraction model for bibliometric data analysis based on word embeddings and clustering. *IEEE Access*. 2019;7:145614-145633.
24. Djaballah KA, Boukhalfa K, Boussaid O. Sentiment analysis of Twitter messages using word2vec by weighted average. Paper presented at: Proceedings of the Sixth International Conference on Social Networks Analysis; 2019:223-228; IEEE, New York, NY.
25. Koppel M. Automatically categorizing written texts by author gender. *Lit Ling Comput*. 2002;17:401-412.
26. Stamatatos E. A survey of modern authorship attribution methods. *J Am Soc Inf Sci Technol*. 2009;60(3):548-556.
27. Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., Xu, K. Adaptive recursive neural network for target-dependent twitter sentiment classification. Proceedings of the 52nd Annual Meeting of Association of the Computational Linguistics. Jule 2014:49-54; ACL, New York, NY.
28. Pennington J, Socher R, Manning C. Glove: global vectors for word representation. Paper presented at: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing; October 2014:1532-1543.
29. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information; 2016. arXiv preprint.
30. Mixing Dirichlet topic models and word embeddings to make lda2vec. https://www.datacamp.com/community/tutorials/lda2vec-topic-model. Accessed January 30, 2020.
31. Arora S, Liang Y, Ma T. A simple but tough to beat baseline for sentence embeddings. Paper presented at: Proceedings of ICLR 2017; April 2017:1-4; ACL, New York, NY.
32. LeCun Y. *Generalization and network design strategies*. Amsterdam, Netherlands: Elsevier; 1989:1-24.
33. Elman JL. Finding structure in time. *Cognit Sci*. 1990;14(2):179-211.
34. Zhang L, Wang S, Liu B. Deep learning for sentiment analysis: a survey. *Data Mining Knowl Discov*. 2018;8(4):1-12.
35. Rojas-Barahona LM. Deep learning for sentiment analysis. *Lang Linguist Compass*. 2016;10(12):701-719.
36. Cho K, Van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
37. Wang, J., Liu, T.W., Luo, X., Wang L. An LSTM approach to short text sentiment classification with word embeddings. Paper presented at: Proceedings of ROCLING 2018; October 2018:214-223; ACL, New York.
38. Aydoğan M, Karcı A. Improving the accuracy using pretrained word embeddings on deep neural networks for Turkish text classification. *Phys A Stat Mech Appl*. 2020;541:123288.
39. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res*. 2014;15(1):1929-1958.

How to cite this article: Onan A. Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks. *Concurrency Computat Pract Exper*. 2020;e5909. https://doi.org/10.1002/cpe.5909