Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 3, 2026

WORD EMBEDDING-BASED NEURAL NETWORKS' PERFORMANCE ON LOW-RESOURCED LANGUAGE

Zubair Uddin[*1], Muhammad Ameen Chhajro[2], Farhan Bashir Shaikh[3], Muhammad Hibatullah Channa[4], Fakhira Tabassum[5], Dr. Kirshan Kumar Luhana[6], Adnan Jahangir Panhwar[7]

*[*1,2]Department of Software Engineering, Sindh Madressatul Islam University, Karachi*
[3]Department of Computer Science, University of Larkano
[4]Department of Computer Sciences & Related Studies, Hyderabad Institute for Technology & Management Sciences, Hyderabad
[5]Department of Computer Science, NUML University, Hyderabad campus
[6]Department of Computer Science, Sindh University Laar campus, Badin
[7]Department of Computer Science, The University of Sindh, Jamshoro

[*1]zubair@smiu.edu.pk, [2]ameen.chhajro@smiu.edu.pk, [3]farhan@uolrk.edu.pk, [4]hibatullah@hitms.edu.pk, [5]fakhira.tabassum@numl.edu.pk, [6]kirshan.luhano@usindh.edu.pk, [7]adnanpanhwar1@gmail.com

DOI: https://doi.org/10.5281/zenodo.19606633

Keywords
Word Embeddings, Word2Vec Skip-gram, LSTM Neural Network, Sindhi Language Corpus, Natural Language Processing

Article History
Received: 22 Jan 2026
Accepted: 14 March 2026
Published: 30 March 2026

Corresponding Author: *
Zubair Uddin

Abstract
Word embedding is a key concept in deep learning, particularly in the realm of natural language processing (NLP). It enables the efficient and accurate representation of words or phrases in vector spaces, capturing the contextual and semantic relationships between them. In this article we have applied different word embedding techniques, comparison analysis on the Sindhi language corpus, and proposed the word embedding framework for the Sindhi Language. The word embedding techniques discussed and applied in this research are Bert-base-uncased, word2vec (CBOW and Skip-gram) on an LSTM neural network. In the end, we evaluate word embeddings' performance on the low-resourced Sindhi Language corpus.

1. Introduction

In the field of Natural Language Processing (NLP), the performance of word embedding-based LSTM neural networks has shown promise even in scenarios where resources for a specific language are limited. Low-resource languages pose unique challenges due to a scarcity of labelled data, making it difficult to build accurate NLP models. However, it is crucial not to overlook the significant impact of ANNs in NLP[1]. Across a variety of language-related activities, these networks have demonstrated

outstanding effectiveness. From machine translation, named entity recognition, and question-answering systems to text categorization and sentiment analysis[2]. The ANNs have enabled machines to comprehend and generate human-like language with impressive proficiency. However, leveraging word embeddings and employing specific strategies can enhance the performance of such models in low-resource language settings. This article explores how word embedding-based LSTM neural networks perform in low-resource Sindhi language and highlights key techniques to improve their efficacy. LSTM (Long Short-Term Memory) is a specialized type of recurrent neural network (RNN) architecture that tackles the challenge of capturing long dependencies in sequential data. It achieves this by introducing a memory unit and a gate mechanism, which work together to determine how to utilize and update the information stored in the memory cell [3]. One of the primary challenges in NLP is the representation of words in a manner that can be easily understood and manipulated by machine learning algorithms. Traditional approaches, such as one-hot encoding, suffer from the curse of dimensionality and fail to capture semantic relationships between words. This is where word embeddings come into play, addressing these issues by providing dense vector representations of words. In this research, we propose a word embedding based (Word2Vec) model using LSTM neural network for Sindhi Language.

1.2 Problem statement

Text data preprocessing and analysis have always been complex tasks in the field of Natural Language Processing (NLP) and Information Retrieval [4]. This complexity extends to languages like Sindhi, which have a rich source of text data, presenting challenges for researchers in this intelligent technology domain. When analysing text for NLP tasks, one crucial step is creating a feature vector. The quality of the feature vector plays a significant role, as a good feature vector leads to good results, while a poor one can result in disastrous outcomes. Unfortunately, traditional methods such as the one-hot model or vector space model for feature extraction fall short in handling the complexity of such text. Mikolov in [5] has revolutionized the field of word representation learning. It provides an efficient and powerful approach to generate word embeddings that facilitate semantic understanding and enable NLP models to leverage the rich relationships between words. One notable limitation is their lack of consideration for contextual information. Traditional models treat words as independent features, overlooking the important context in which they appear. Consequently, they struggle to capture the subtle nuances and meanings that emerge from the interactions and dependencies between words within a sentence or document[6]. Another drawback is the issue of data sparsity. Traditional models often represent text using sparse representations such as one-hot encoding or TF-IDF, resulting in high-dimensional feature spaces with many empty dimensions. This sparsity hampers their effectiveness in capturing meaningful patterns and relationships.

1.3 Research Contribution

- We have prepared the Sindhi language corpus, which contains 2.4M Vocabulary.
- We have trained the word embedding network on a language corpus and monitor its performance.
- Comparison analysis that how Neural networks perform on low-resourced language like Sindhi.

2. Background Study

2.1 Neural Networks for Natural Language Processing

The Neural network also called as the Artificial Neural Networks (ANNs), which are the sub part of the machine learning. Their structure has been designed from the concept of the human brain, as they communicate with each other through signals [7]. The neural network is composed of the interconnected nodes that makes the structure of neural networks, it has input layer, in middle it has one or more hidden

layers and an output layer. Each node, or artificial neuron, is connected to others and has a weight and threshold that go along with it. Any node whose output exceeds the defined threshold(bias) value then a neuron is activated and begins providing data to the network's next layer. Otherwise, no data is transmitted to the network's next layer. The function depends on parameters i.e., weights and bias values.

In a standard neural network (NN), the fundamental building blocks are the neurons. These neurons are interconnected processors that work together to process information. Each neuron produces a sequence of real-valued activations, which are numerical values representing its level of activity[8]. For predictions neural networks use mathematical functions for computation purposes, here, the function f (x) represents the output of the neuron, for x inputs, w weights, and the threshold (bias) value. the basic function can be seen in equation1 below:

$$Y = f(\sum(W * X) + b), \dots\dots\dots(1)$$

A neural network is one of the popular algorithm types in machine learning that covers the broad range of concepts and techniques, and it has been designed to simulate the way the human brain works[9]. However, sometimes we call them black boxes, because it is difficult to understand what is going on inside and what they are doing. The neural network is made of layers of interconnected nodes that work together to process and analyse the complex form of the data, like, text, speech, and images. Each node in the network takes the information from the other nodes which are connected to it, that process that information, and the passes the output on to the next layer of nodes[10]. The neural network is trained on the set of data, using a process or method called backpropagation, this method supports the neural network to learn how to make accurate predictions or classifications based on new or updated data it expects in the future. However, we cannot just fit a straight line to the data make predictions, so note the point we can use a straight line to make future predictions, we us the equation of regression line also called best fit line. linear regression is one the Easiest and most popular supervised machine learning algorithms used to find relationship between the model predictions[11] .it is a Statistical method that is used to make predictions for continuous numeric values. For better understanding for continuous values like, sale, salary, price, age.

3. Proposed Framework

In this research, we have applied one of the popular words embedding techniques Word2Vec to train the Long Short-Term Memory (LSTM) model of Recurrent Neural Networks (RNNs). Our workflow, presented in Figure 1, consists of several steps to train a deep learning-based framework using a corpus of Sindh Language. To begin, we load and process the source code files, extracting relevant information to create sequence data along with their corresponding labels. This initial stage involves preparing the data for further analysis. Subsequently, the generated sequence data goes through a transformation process in the next stage. We convert the code snippets into code embedding vectors in this case, it is Word2Vec, which captures the meaning and syntax of the code in a numerical representation. Once the code embedding vectors are prepared, they are partitioned and used as input for the constructed neural networks during the training phase. LSTM model, utilizing the word embedding technique, processes these vectors to learn and understand the patterns and relationships within the code. After the training process is completed, we evaluate the performance of the trained model by testing its

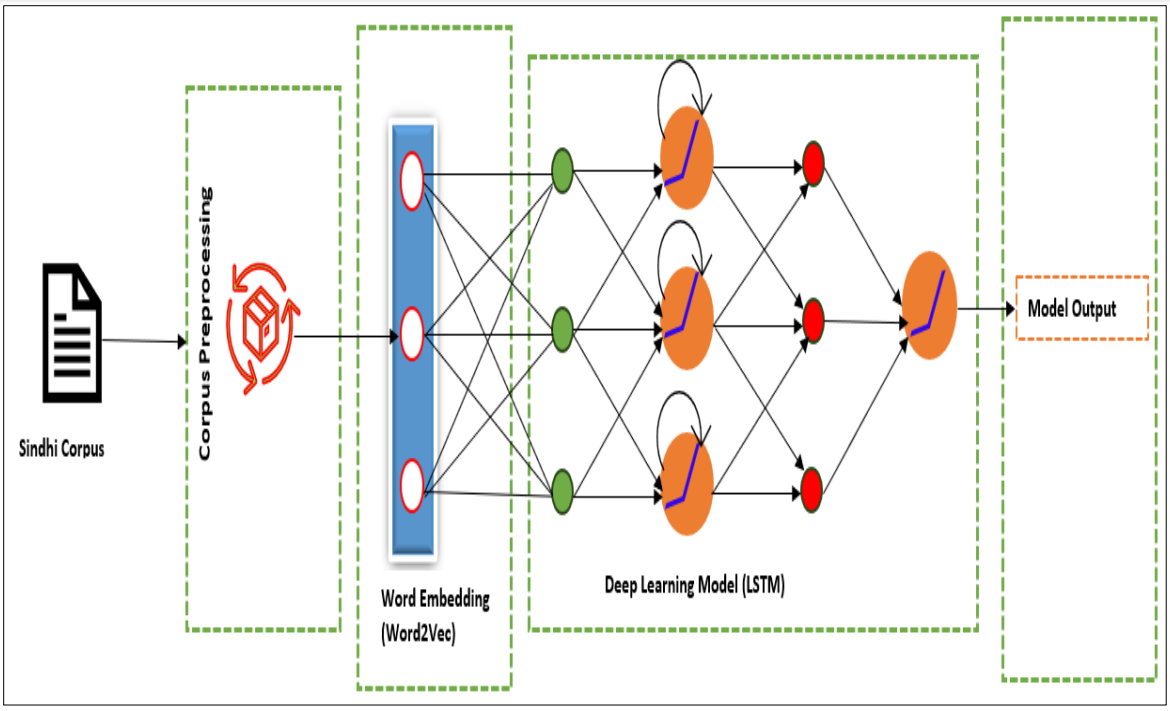

Figure 1. Proposed Word Embeddings-based Framework for Sindhi Language

accuracy and error. Additionally, we collect detailed logs automatically during this testing phase to gain insights into how the models perform and behave in different scenarios, such as changing the number of epochs during the training. This section has been further broken down into small sections as per the proposed research.

3.1 Dataset Development

The development of text corpus for Sindhi Language involves creating a collection of textual data which has been collected from various sources with the help web scraping techniques. In this research work, we have developed the language corpus from scratch, text normalization and data analysis.

3.2 Data Collection and Corpus Preparation

As a complete automatic pipeline's accuracy from beginning to end is highly dependent on the generated corpus into a resource for linguists and academics[12]. As per knowledge no standard collections of Sindhi text data are utilized for business and natural language processing applications. The data gathered from multiple sources and making it clean by eliminating garbage is also one of the big challenges in the data preparation. The Sindhi corpus has been prepared from the Newspapers, Books, Twitter, and other online web sources through the web scraping techniques. The details of developed corpus have been shown in Table 1. Although there are no pre-defined criteria for how the corpus should be constructed, having sufficient, accurate, and representative data gives a suitable outcome.

The deep learning technique requires different inputs and a sufficient pre-processing step to remove noise[13].The google Colab has been used as a development environment and python as a language for Sindhi text preprocessing.

Table 1. Sindhi Language Corpus details

Corpus Categories	Sentences	Word Tokens
Awami awaz	32883	
Wikipedia	175145	**2.4 M**
Books	128625	
Kawish	205618	
Twitter	11571	
Other Sources	87552	

4. Neural Word Embeddings

The Neural Word Embedding Generator is a tool or algorithm that takes a collection of texts T, known as a corpus, as input. Its purpose is to produce a set of feature vectors called word embeddings E, where each vector represents the embedding of a particular word found in the corpus[14]. To generate word embeddings, the Neural Word Embedding Generator typically employs neural network models such as FastText[15] ,word2vec[16], GloVe[17], and BERT. Word embedding has become an integral part of many NLP tasks such as language translation, text classification, sentiment analysis, and more. These techniques enable the capture and representation of important semantic and syntactic properties within code [18].

4.1 Word2Vec Word Embedding

Word2Vec is a widely used technique for generating word embeddings used for neural networks to take their input word representation in the form of vectors [19], which are continuous vector representations of words derived from large text corpora. One of the research in [20] has discussed the parameters of Word2Vec, their evaluation, and the architectures of CBOW and Skip-Gram in detail. The Word2Vec model utilizes two evaluation methods: Hierarchical Softmax and Negative Sampling. Hierarchical Softmax was initially introduced by Morin and Bengio [21]. The intermediate nodes in the tree explicitly encode the relative probabilities of their child nodes. By employing this hierarchical structure, Hierarchical Softmax enables efficient estimation of word probabilities during training. On the other hand, Negative Sampling provides a simpler alternative to Hierarchical Softmax [22]. Instead of using a complex binary tree, Negative Sampling selects a subset of output words as negative samples and updates them during training. These negative samples are randomly chosen, allowing for computational efficiency while still providing valuable training signals. There are two main categories of Word2Vec embeddings: Continuous Bag-of-Words (CBOW) and Skip-gram.

4.2 Skip-Gram Word Embedding

The Continuous Bag-of-Words (CBOW) approach focuses on predicting a target word based on its surrounding context words. It considers a window of neighboring words and attempts to predict the word in the center. CBOW embeddings are efficient to train and work well when words are frequent in occurrence, without much emphasis on their positional significance within the context. In contrast, the Skip-gram approach reverses the CBOW process. It aims to predict the context words given a target word. By taking a target word, the model tries to forecast the words that usually appear in its vicinity[23]. Skip-gram embeddings excel at capturing semantic information and are effective even with less common words. However, training Skip-gram models generally takes more time compared to CBOW. An example of focused and context-based words can be seen in figure 2.

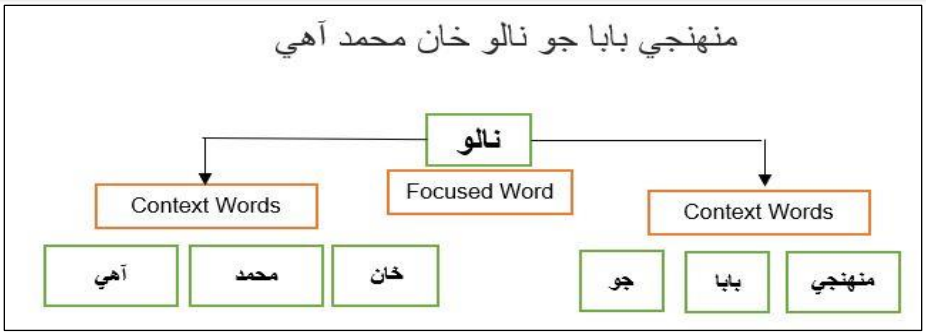

Figure 2. Focused and Context Words in Skip-gram.

In this example, we have taken a Sindhi language sentence, and it has been divided into context words and one focused word. The window size of context words in this example is three (03), meaning we are considering the three words from each side of the focused word.

The skip-gram model predicts the surrounding words given the target word. In this context, "skip-gram" refers to a window of words surrounding a focus word, with the fourth word in the given sentence being the focus word. The Context-independent Skip-Gram word-embedding proposed method's schematic diagram is given in figure 4. In other words we interested to evaluate the parameters meaning the vector presentations of both focused and contexts words, such that we take the dot product of both Vw.Vc associated if the value of dot product is "good" meaning word-context pairs is maximized[24].Both CBOW and Skip-gram models employ Word2Vec algorithms, which rely on shallow neural networks with a single hidden layer. During training, the hidden layer's weights, which serve as the word embeddings, are iteratively learned. These embeddings provide compact vector representations of words, enabling algorithms to understand and exploit the relationships and similarities between words based on their distribution patterns in the training data. So, both model's beliefs on the very common rule of "know a word by the company it keeps" [25].

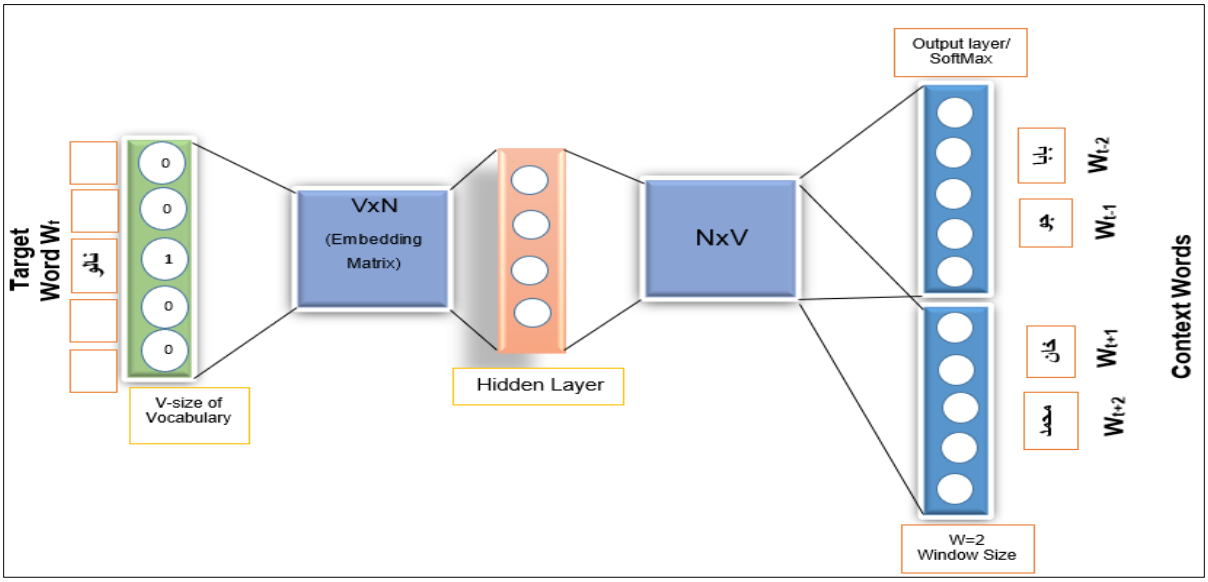

Figure 3. Skip-gram word-embedding method

In Skip-Gram method above there is input text sentence from Sindhi Language and it has one target word Wt. The input text of whole data with vocabulary size V is converted into one hot vector which is in the form of 0's and 1's. In result we have VxN embedding matrix of text data. The there is one hidden layer which uses the activation function gives the NxV that is transpose matrix of features. The output of hidden layer becomes the input of Softmax layer to decide about best possible context words with respect to input target word in Skip-Gram model.

4.3 Continuous Bag-of- Words (CBOW)

The CBOW predicts the probability of a word given the surrounding words [26]. Each word is coded in One-hot form. One hot representation of one sentence from Sindhi Corpus and example of focused and context words of CBOW can be seen in Figure 5.

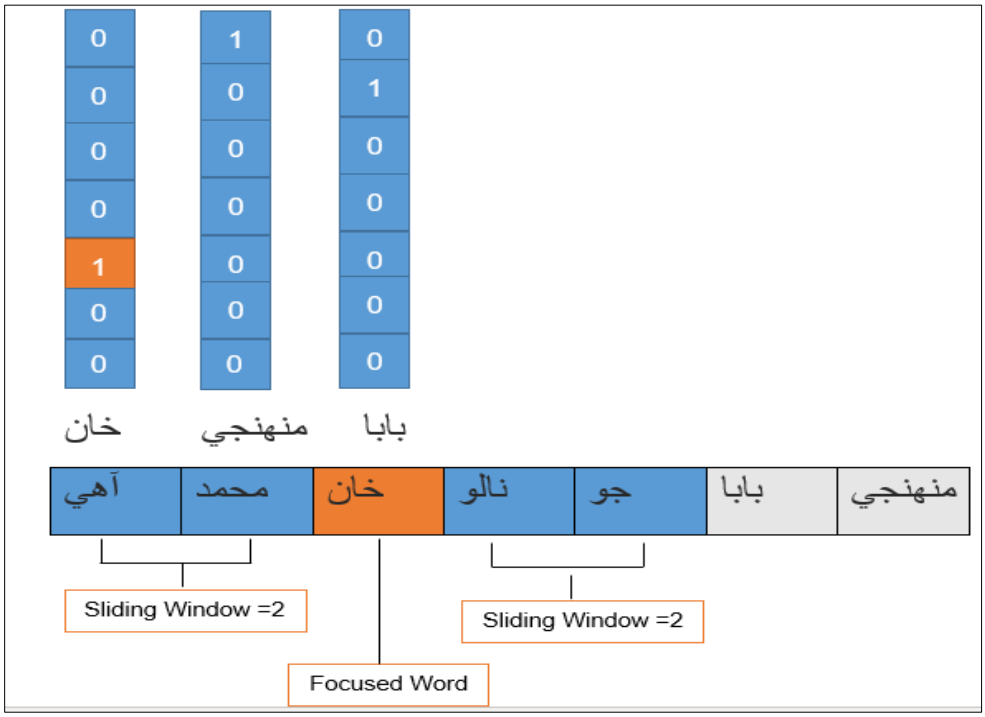

Figure 4. Focused and Context words in CBOW

A sliding window of context words. Single hidden and output layer. Corresponding elements are set to 1 and all elements are zero. As shown in Figure 4, CBOW predicts the target word which is (خان) in this case by considering the surrounding context words.

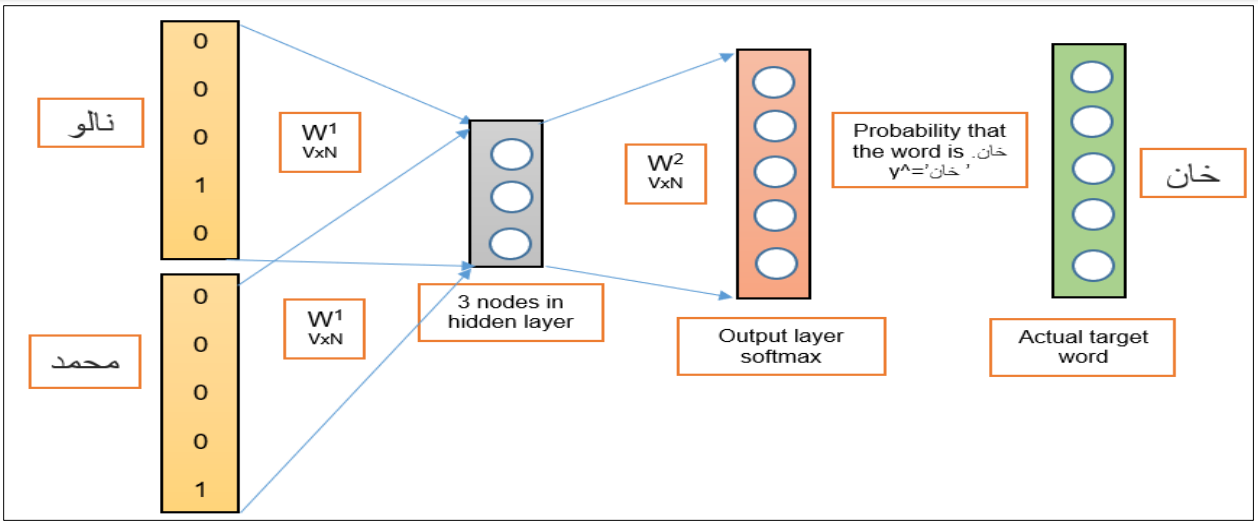

Figure 5. CBOW Word Embedding Method.

The CBOW model predicts the target word by examining the surrounding terms within a specified window, BOW can be explained mathematically as, W is the weight matrix, generates one hot vectors of window size m. The input one hot vectors or context can be represented as in the equation2 given bellow,

$$X^c = (X^{(c-1)}, \dots X^{(c-m)}, X^{(c+1)}, \dots X^{(c+m)} \dots (2)$$

To get embedded word vector for context can be represented mathematically as follows,

$$V^{c+1} = (W1 * X^{(c+1)} \dots (3)$$

$$V^{c-1} = (W1 * X^{(c-1)} \dots (4)$$

$$V^{c-m} = (W1 * X^{(c-m)} \dots (5)$$

Mathematical representation of average vectors is,

$$(V)^{ave} = \left(\frac{V^{(c-m)} + V^{(c-m+1)} + V^{(c+m-1)} + V^{(c+m)}}{2m} \right). (6)$$

To generate the Z score we can define it through the following equation 7.

$$Z = \mathrm{W}^2 * V^{ave}, \dots \dots \dots \dots \dots \dots \dots (7)$$

In the end, we calculate the probability, that may be represented by Y^ by using the softmax and z score we have calculated before in the equation 7. Therefore, probability to calculate the target word by the surrounding words will be as,

$$\mathrm{Y}^{\wedge} = Softmax(Z) \dots \dots \dots \dots \dots \dots (8)$$

4.4 RNN-based Context-aware Word Embedding

In a bidirectional RNN model, the hidden states of the forward and backward RNNs are typically combined or concatenated to create a representation that incorporates information from both directions. However, word embedding itself does not directly connect the hidden states of the forward and backward RNNs. Instead, it is used as input to the RNNs, providing a meaningful representation of words for the model to process [27].

5. Long Short-Term Memory (LSTM) Network

The Long Short-Term Memory, or LSTM, is a special kind of neural network known for its ability to work with sequential data like sentences, time series, or speech [28]. It was developed to address a common issue faced by traditional recurrent neural networks (RNNs) when dealing with long sequences of information. In normal RNNs, a problem called the "vanishing gradient problem" arises and discussed in [29], making it difficult for the network to learn from distant past information. LSTM was designed to tackle this problem by introducing memory cells and gating mechanisms that enable it to remember or forget specific information over time. The main components of an LSTM are the input gate, forget gate, cell state, output gate, and hidden state. Each gate has a specific role in managing the information flow. The input gate decides what information from the current input should be stored in the memory cell. The forget gate determines which information from the previous cell state should be forgotten. The output gate controls which information from the cell state should be passed on as the output [30]. The cell state is like the brain of the LSTM, storing essential information from previous time steps. The hidden state is the output that carries relevant information to the next time step or to the final output layer. Mathematically, LSTM uses various equations involving sigmoid and tanh activation functions to perform its tasks [31]. These equations calculate the values for the input gate, forget gate, output gate, and candidate cell state, which is used to update the cell state. Thanks to its architecture, LSTM networks can handle long sequences effectively, making them ideal for natural language processing, speech recognition, and time series predictions. However, they are computationally intensive and can overfit if not properly tuned and regularized [32]. Therefore, like any neural network, successful implementation of LSTM depends on careful data preprocessing, thoughtful architecture design, hyperparameter tuning, and appropriate training. LSTM has been instrumental in various fields, revolutionizing natural language understanding, machine translation, and speech recognition, among others.

6. Results and Discussions

BERT works for word representation in terms of their context and it is based on transformers, which can be modified for specific tasks[33]. In this research we have applied different embedding models like BERT, the BERT that has been applied in this research is a pre-trained(bert-base-uncased) using pytorch and tensor flow and max input sequence length is 128.For BERT we generated the input ids and attention masks for BERT using Pytorch. Then we trained the BERT model on Sindhi text data by applying the pre-trained BERT (bert-base-uncased). The word embedding result of Bert has been measured through the similarity, and the Bert similarity for Sindhi data is 73.23%, as we can see in the table 2 below.

Table 2. Comparison analysis of word embedding models on Sindhi language corpus

Input Data	Embedding Models	Testing Words	Similarity Results
628695	Bert-base-uncased	kingdom =باشاهي Authority= حاڪميت	73.28%
628695	Word2Vec (CBOW)	Universities= يونيورسٽين Colleges= ڪاليجن	87.2%
628695	Word2Vec (Skip-gram)	Universities= يونيورسٽين Colleges= ڪاليجن	84.84%

To compare the performances, we have also used Word2Vec with its both techniques, we obtained the 87.28% similarity result by Pre-Trained Word2Vec (CBOW), and 84.84% from the Pre-Trained Word2Vec (Skip-Gram). The Pre-Trained Word2Vec (CBOW) performed well on Sindhi corpus.

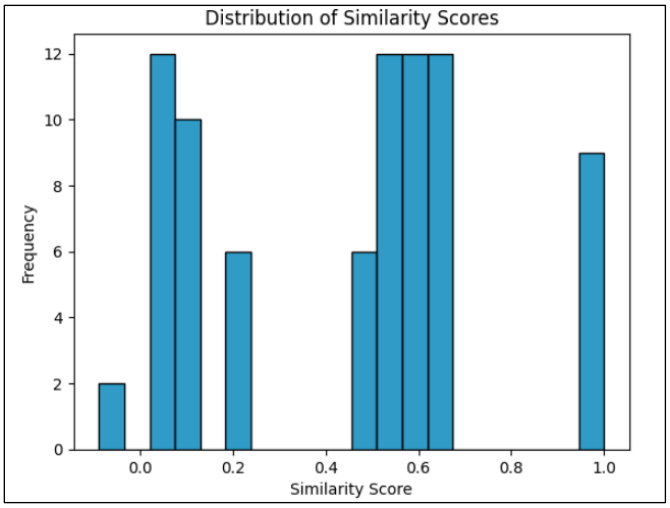

Figure 6. BERT (bert-base-uncased) Similarity Score

Similarity result of pre-trained(bert-base-uncased) can be seen in the figure 6 above. It shows the similarity score distribution between the word's frequency and similarity score.

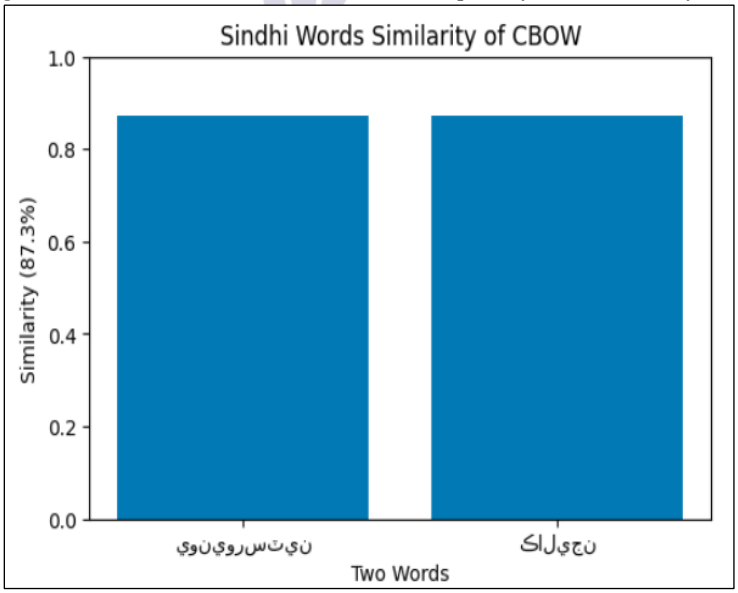

Figure 7. Similarity result of CBOW between two words

Despite these differences, CBOW and Skip-gram share common ground. They both use a neural network architecture and are trained on large amounts of text data. We can see the performance of CBOW in Figure 7 above. A CBOW similarity heatmap of text data provides a visual representation of how words are semantically related to each other, as learned by the CBOW word embedding model. It's like a color-coded chart that showcases the similarity

between different words. As we can see from the similarity heatmap in the figure 8 below.

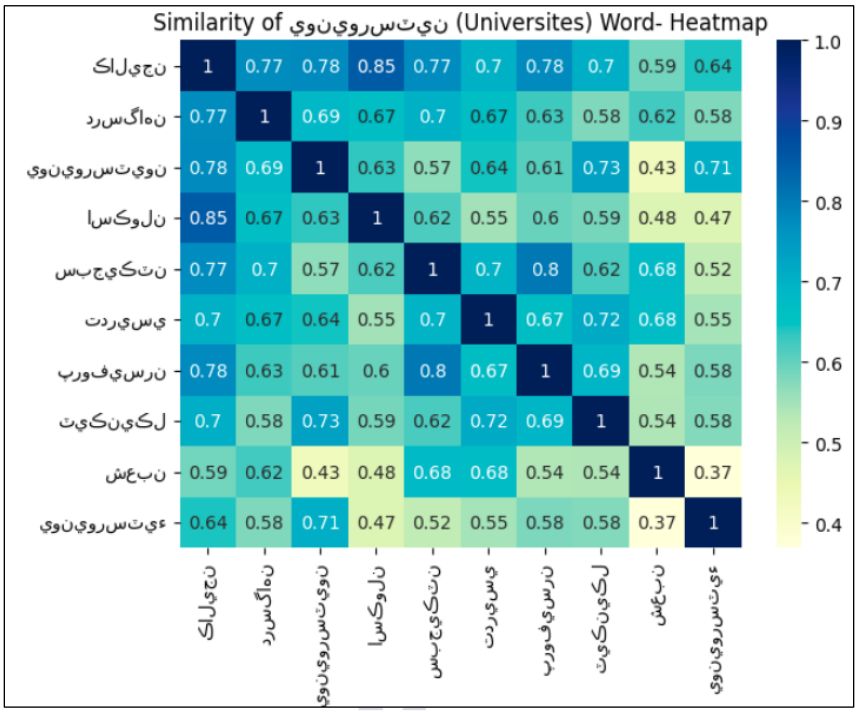

Figure 8. Heatmap-Similarity of word Universities in Sindhi Corpus

The underlying idea for both algorithms CBOW and Skip-Gram is the distributional hypothesis, which suggests that words with similar meanings tend to appear in similar contexts. As, for Skip-Gram model we can see it's result in figure 9 below which finds the similarity between two Sindhi words from the corpus.

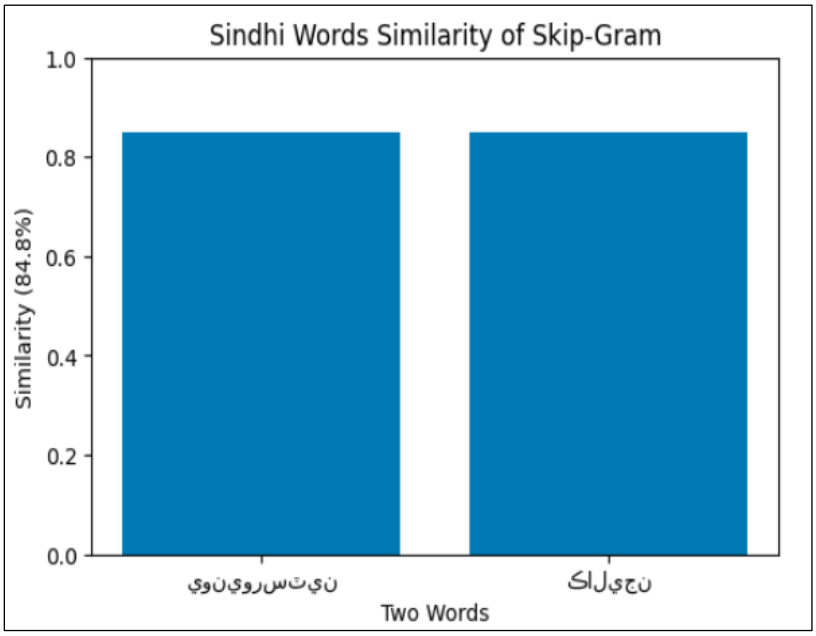

Figure 9. Similarity result of Skip-Gram

We analysed the LSTM performance on embeddings of the Sindhi corpus, in terms of training and validation accuracy and training and validation. As a result, the LSTM accuracy can be seen in Figure 10.

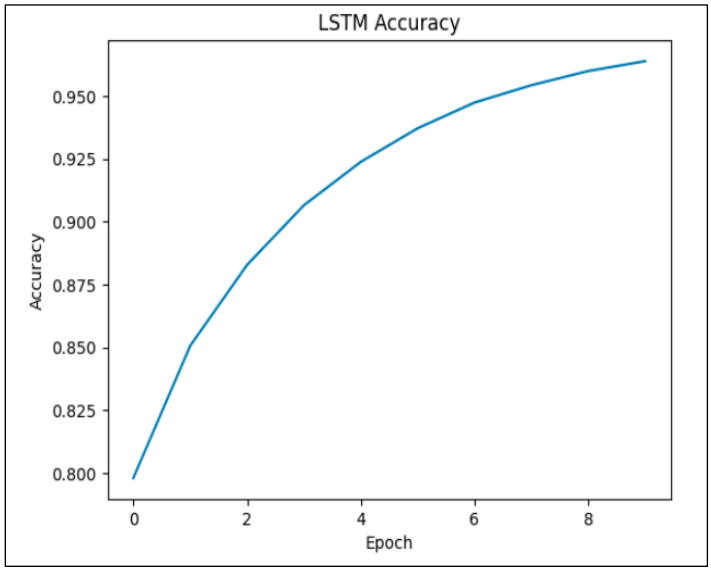

Figure 10. LSTM Accuracy result on word embedding.

Similarly, we can see the loss result of LSTM on word embeddings of Sindhi Corpus in figure 11 given below.

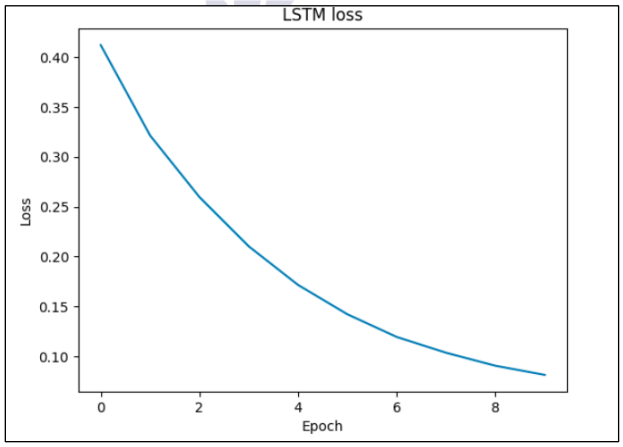

Figure 11. LSTM Loss on word embeddings.

When we trained an LSTM model for working with text, we evaluated its performance using two types of accuracy: training accuracy and validation accuracy. As we can check LSTM performance in table 3 given below.

Table 3. LSTM performance on word embeddings of Sindhi Corpus

Model	Accuracy & Loss	Results
Long Short-Term Memory (LSTM)	Training Accuracy	**96.38%**
	Validation Accuracy	**81.15%**
	Training Loss	**0.0815**
	Validation Loss	**0.6808**

The training accuracy shows us how well the model has learned from the data it was trained on - that is, the text corpus used during the training process. As the model learns and improves its predictions, the training accuracy increases because it becomes more accurate in making predictions on the data it has seen before. The same we can see our model is performing well during training. On the other hand, validation accuracy helps us understand how well the model generalizes to new, unseen data. To do this, we set aside a separate dataset called the validation set, which the model has never encountered during training. This allows us to test how the model will perform in real-world situations, dealing with text it hasn't learned from before.

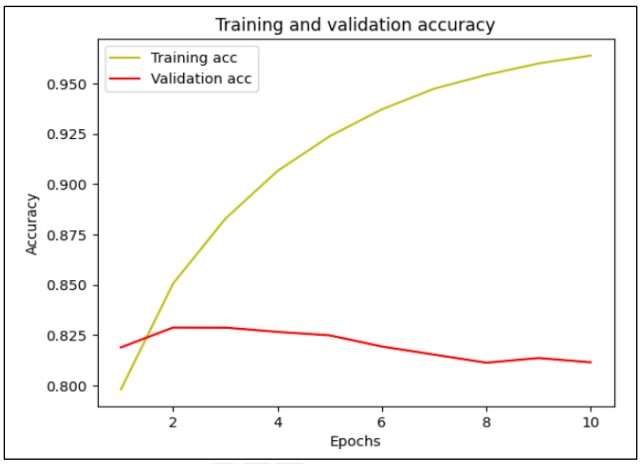

Fig. 12. LSTM Training and Validation Accuracy on word embeddings

So, there is no big difference between the two accuracy results, and we can conclude that our model performance is good on Sindhi word embeddings. The comparison analysis to both training and validation accuracy of word embeddings on the Sindhi language corpus can be seen in Figure 12 above. When we have both the accuracies of LSTM, we can see the result of losses, the training loss, and the validation loss of the model on word embeddings. generated from the Sindhi language corpus. The result can be seen in figure 13 below.

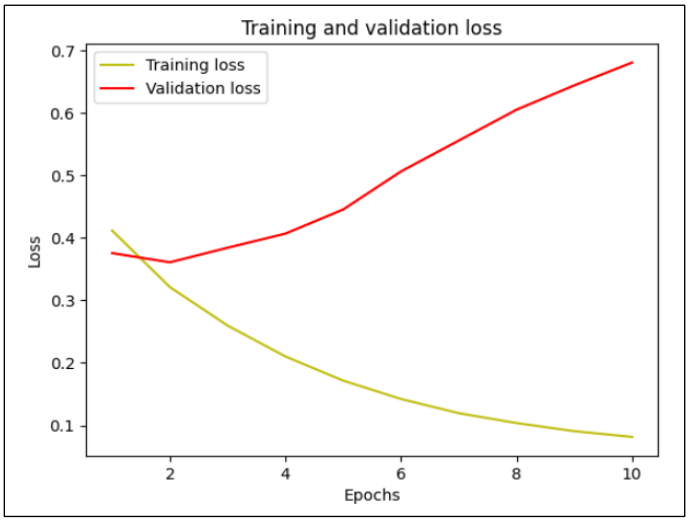

Figure 13. LSTM Training and Validation Loss on word embeddings

8. Conclusion

This research has implemented an LSTM model and used the pre-trained BERT and Word2Vec embeddings as the initial weights of the embedding layer. This study analysed the performance of Sindhi word embeddings and the combination of Word2Vec and LSTM enhanced our ability to handle Sindhi language by effectively capturing contextual information and semantic relationships in the text data. As a result, we can utilize this Language embeddings on a wide range of Sindhi language and natural language processing tasks.

9. REFERENCE

B. A. Chandio, A. S. Imran, M. Bakhtyar, S. M. Daudpota, and J. Baber, "Attention-Based RU-BiLSTM Sentiment Analysis Model for Roman Urdu," *Appl. Sci.*, vol. 12, no. 7, 2022, doi: 10.3390/app12073641.

A. Ligthart, C. Catal, and B. Tekinerdogan, *Systematic reviews in sentiment analysis: a tertiary study*, vol. 54, no. 7. Springer Netherlands, 2021. doi: 10.1007/s10462-021-09973-3.

G. Rao, W. Huang, Z. Feng, and Q. Cong, "LSTM with sentence representations for document-level sentiment classification," *Neurocomputing*, vol. 308, pp. 49–57, 2018, doi: 10.1016/j.neucom.2018.04.045.

V. Gurusamy and Kannan S, "Preprocessing Techniques for Text Mining," *Int. J. Comput. Sci. Commun. Networks*, vol. 5, no. 1, pp. 7–16, 2014.

T. Demeester, T. Rocktäschel, and S. Riedel, "Lifted rule injection for relation embeddings," *EMNLP 2016 - Conf. Empir. Methods Nat. Lang. Process. Proc.*, pp. 1389–1399, 2016, doi: 10.18653/v1/d16-1146.

W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, no. October 2016, pp. 11–26, 2017, doi: 10.1016/j.neucom.2016.12.038.

I. C. Education, "Neural Networks," 2020. https://www.ibm.com/cloud/learn/neural-networks (accessed Oct. 27, 2022).

Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

R. Sun, "tngineering," vol. 5, 1990.

J. Bossart, "Opening the black box," *Pharm. Manuf. Pack. Sourcer*, no. WINTER, pp. 32–34, 2010, doi: 10.5860/crl.77.5.564.

D. Maulud and A. M. Abdulazeez, "A Review on Linear Regression Comprehensive in Machine Learning," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 4, pp. 140–147, 2020, doi: 10.38094/jastt1457.

M. Meelen, É. Roux, and N. Hill, "Optimisation of the Largest Annotated Tibetan Corpus Combining Rule-based, Memory-based, and Deep-learning Methods," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 20, no. 1, pp. 1–11, 2021, doi: 10.1145/3409488.

R. G, "A Study to Find Facts Behind Preprocessing on Deep Learning Algorithms," *J. Innov. Image Process.*, vol. 3, no. 1, pp. 66–74, 2021, doi: 10.36548/jiip.2021.1.006.

D. Dessì, M. Dragoni, G. Fenu, M. Marras, and D. R. Recupero, "Evaluating neural word embeddings created from online course reviews for sentiment analysis," *Proc. ACM Symp. Appl. Comput.*, vol. Part F1477, pp. 2124–2127, 2019, doi: 10.1145/3297280.3297620.

A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText.zip: Compressing text classification models," pp. 1–13, 2016, [Online]. Available: http://arxiv.org/abs/1612.03651

T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, pp. 1–12, 2013.

HANSON ER, "Musicassette Interchangeability. the Facts Behind the Facts," *AES J. Audio Eng. Soc.*, vol. 19, no. 5, pp. 417–425, 1971.

H. N. Nguyen, S. Teerakanok, A. Inomata, and T. Uehara, "The comparison of word embedding techniques in RNNS for vulnerability detection," *ICISSP 2021 - Proc. 7th Int. Conf. Inf. Syst. Secur. Priv.*, no. Icissp, pp. 109–120, 2021, doi: 10.5220/0010232301090120.

E. Biswas, K. Vijay-Shanker, and L. Pollock, "Exploring word embedding techniques to improve sentiment analysis of software engineering texts," *IEEE Int. Work. Conf. Min. Softw. Repos.*, vol. 2019-May, no. May, pp. 68–78, 2019, doi: 10.1109/MSR.2019.00020.

P. F. Muhammad, R. Kusumaningrum, and A. Wibowo, "Sentiment Analysis Using Word2vec and Long Short-Term Memory (LSTM) for Indonesian Hotel Reviews," *Procedia Comput. Sci.*, vol. 179, no. 2020, pp. 728–735, 2021, doi: 10.1016/j.procs.2021.01.061.

F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," *AISTATS 2005 - Proc. 10th Int. Work. Artif. Intell. Stat.*, pp. 246–252, 2005.

C. Besana, M. Memoli, P. M. Salvioni, R. A. Finazzi, F. Inversi, and C. Rugarli, "Meperidine in detoxification of hospitalized heroin addicts," *Subst. Use Misuse*, vol. 26, no. 5, pp. 505–513, 1991, doi: 10.3109/10826089109058901.

R. P. Nawangsari, R. Kusumaningrum, and A. Wibowo, "Word2vec for Indonesian sentiment analysis towards hotel reviews: An evaluation study," *Procedia Comput. Sci.*, vol. 157, pp. 360–366, 2019, doi: 10.1016/j.procs.2019.08.178.

B. L. Eppley, "Mentoplasty," *Soft-Tissue Surg. Craniofacial Reg.*, pp. 351–358, 2007, doi: 10.1097/00006534-198205000-00031.

M. M. Louwerse, "Knowing the Meaning of a Word by the Linguistic and Perceptual Company It Keeps," *Top. Cogn. Sci.*, vol. 10, no. 3, pp. 573–589, 2018, doi: 10.1111/tops.12349.

H. Ahmed Chowdhury, M. A. Haque Imon, and M. S. Islam, "A Comparative Analysis of Word Embedding Representations in Authorship Attribution of Bengali Literature," *2018 21st Int. Conf. Comput. Inf. Technol. ICCIT 2018*, pp. 1–6, 2019, doi: 10.1109/ICCITECHN.2018.8631977.

F. M. Yajie Miao, Mohammad Gowayyed and Language, "EESEN : END-TO-END SPEECH RECOGNITION USING DEEP RNN MODELS AND WFST-BASED DECODING Yajie Miao , Mohammad Gowayyed , Florian Metze," *Proc. ASRU*, pp. 167–174, 2015.

Y. Zhu *et al.*, "What to do next: Modeling user behaviors by Time-LSTM," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 0, no. March 2018, pp. 3602–3608, 2017, doi: 10.24963/ijcai.2017/504.

Y. Wang, Z. Gao, M. Long, J. Wang, and P. S. Yu, "PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 11, pp. 8122–8131, 2018.

B. Xing *et al.*, "Earlier attention? Aspect-aware LSTM for aspect-based sentiment analysis," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2019-Augus, pp. 5313–5319, 2019, doi: 10.24963/ijcai.2019/738.

A. Farzad, H. Mashayekhi, and H. Hassanpour, "A comparative performance analysis of different activation functions in LSTM networks for classification," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 2507–2521, 2019, doi: 10.1007/s00521-017-3210-6.

S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, 2018.

H. N. Dang, K. Lee, and S. Henry, "Ensemble BERT for classifying medication-mentioning Tweets," no. Figure 1, pp. 37–41, 2018.