CANBLWO: A Novel Hybrid Approach for Semantic Text Generation

Abhishek Kumar Pandey
School of Computer Science and Engineering
Vellore Institute of Technology, India
abhishek.pandey2020@vitstudent.ac.in

Sanjiban Sekhar Roy
School of Computer Science and Engineering
Vellore Institute of Technology, India
s.roy@vit.ac.in

Abstract: *Semantic text generation is critical in Natural Language Processing (NLP) as it faces challenges such as maintenance of coherence among texts, contextual relevance, and quality output. Traditional language models often produce grammatically inconsistent text. To address these issues, we introduce Convolutional Attention Bi-LSTM with Whale Optimization (CANBLWO), a novel hybrid model that integrates a Convolutional Attention Network (CAN), Bidirectional Long Short-Term Memory (Bi-LSTM), and Whale Optimization Algorithm (WOA). CANBLWO aims to generate semantically rich and coherent text and outperforms the traditional models like Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNN), Bi-LSTM, and Bi-LSTM with attention, Bidirectional Encoder Representations from Transformers (BERT), and Generative Pre-trained Transformer 2 (GPT-2). Our model achieved 0.79, 0.78, 0.76, and 0.82 scores in Metric for Evaluation of Translation with Explicit Ordering (METEOR), Bi-Lingual Evaluation Understudy (BLEU), Consensus-based Image Description Evaluation (Ciders), and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics, respectively. The proposed model also demonstrates 97% and 96% accuracy on Wiki-Bio and Code/Natural Language Challenge (CoNaLa) datasets, highlighting its effectiveness against Large Language Models (LLMs). This study underscores the potential capability of hybrid approaches in enhancing semantic text generation*.

Keywords: *Natural language processing, natural language generation, neural network, convolutional attention network, whale optimization algorithm, BERT, large language model*.

Received April 3, 2024; accepted July 23, 2024
https://doi.org/10.34028/iajit/21/4/11

1. Introduction

Natural Language Generation (NLG) has advanced rapidly in the last ten years. It has helped to generate automatic text such as reports, narratives, and Multiple Choice Questions (MCQ). NLG is a challenging task that requires effective handling of grammar, sentence structure, semantic meaning, contextual relevance, and tone or style. Semantically enhanced text generation aims to produce text that is not only grammatically correct but also semantically meaningful. This means that the text should have a clear and coherent meaning and be relevant to the context of the input dataset that has already been generated. Seo *et al*. [42] have proposed a valuable framework for generating text with semantic control by leveraging the power of grammar-based models. This research uses a transformer-based encoder-decoder model. Yang *et al*. [58] have introduced an innovative approach called dynamic planning for generating text from data. Using dynamic planning techniques, their proposed method addressed the challenges of generating coherent and contextually relevant text. Zhang *et al*. [60] presented emotional text generation, namely cross-domain sentiment transfer, using a Gated Neural Network (GRU)-based encode decoder model.

Semantically enhanced text generation is important in Natural Language Processing (NLP) because it can improve text quality in various ways, such as generating more natural and engaging text. It can also improve the accuracy of machine translation, create more informative, helpful chatbots, and develop more effective marketing content. Shedko [44] proposed an interactive and intelligent system based on Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) for generating creative text. Ding *et al*. [14] presented a text summarization that uses semantic attention to enhance the quality of the text summary generation. The Large Language Model (LLMs) are facing same kind of text generation problem as mentioned by Qu *et al*. [34], in Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer 2 (GPT-2) prediction models have been proposed, and in this work authors have used BERT to generate intermediate words. Here, GPT-2 has been used to generate a long sentence or even articles [34]. LLMs used to go through extensive training steps using massive datasets, enabling them to utilize a spectrum of functions. These include tasks such as translation, generating diverse forms of creative content, and providing informative responses to questions, even when they are complex, unconventional, or puzzling. Therefore, LLMs encounter several hurdles when it comes to generating textual output. In our work, the proposed Convolutional Attention Network (CAN) has enhanced the model's contextual understanding and feature extraction [57]. It has also helped us identify grammatical structures such as verb phrases and entities such as person, organization, or location names. For example, in our work, one instance of the data set, namely Wiki-Bio, is "Otto extra is a German award-winning aerobatic pilot," and the proposed CAN model has identified the feature map as "Otto extra" as a person's name and "German" as the nationality of that person. Bidirectional Long Short-Term Memory (Bi-LSTM) helps us to extract salient features such as co-referencing from the input text; for example, consider a sentence, "extra was trained as a mechanical engineer, and he began his flight training in gliders" here the salient feature "extra" referred to a named entity. Therefore, it is mapped with the pronoun 'he' in the same sentence. Finally, we have adopted the Whale Optimization Algorithm (WOA) for optimizing the hyperparameter tuning [27]. We have combined the advantages of CAN, Bi-LSTM, and WOA to address the above challenges in generating semantically rich text. We also have conducted a comparative analysis of the proposed methodology against popular LLMs such as BERT, GPT2, and other deep learning methods such as Long Short-Term Memory (LSTM), Bi-LSTM, and Bi-LSTM with attention. BERT and GPT2 have demonstrated good results. Moreover, this comparative analysis serves as a valuable benchmark for assessing the performance of the proposed methodology within the context of language models and deep learning frameworks. The generated text by proposed model is evaluated using standard metrics such as Metric for Evaluation of Translation with Explicit Ordering (METEOR), Bi-Lingual Evaluation Understudy (BLEU), Consensus-based Image Description Evaluation (CIDEr), and Recall-Oriented Understudy for Gisting Evaluation (ROUGE). The integration of WOA with the hybrid CAN and Bi-LSTM model presents a promising direction to achieve grammatically improved text.

The contributions of the manuscript are as follows:

- A comprehensive study on the semantic enhancement of text generation problems has been carried out. Existing models for text generation have also been analyzed and how these models have improved the quality of natural and engaging text generation.
- We have proposed a hybrid model CANBLWO that combines a convolutional attention network, Bi-LSTM, and whale optimization. The proposed model has highlighted the importance of the embedding layer, attention layer, Bi-LSTM layer, and dense layer as a stacked hybrid architecture. The proposed model CANBLWO has obtained promising results in comparison with the other existing language models for text generation.
- CAN model has been empowered by Conv1D, and we have shown the extraction process of features from the text dataset such as Wiki-Bio, Code/Natural Language Challenge ConALA, Internet Movie Database (IMDb) and Gigaword. The potential capability of the proposed Bi-LSTM model for the decoding of feature maps of text as a stacking layer has also been demonstrated, and finally, whale optimization has tuned the model in terms of kernel size, neuron, and batch size.

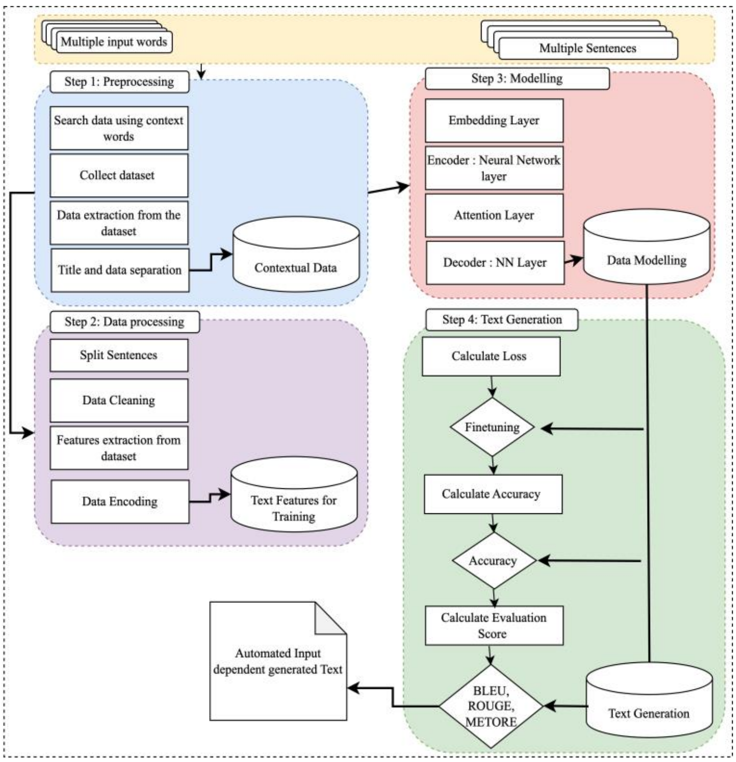


Figure 1. A Generative AI-based approach for context-based text generation.

Figure 1 shows the context-based text generation using generative AI. This model used to operate in four stages: pre-processing, data cleaning, modeling, and text generation. In preprocessing steps, the conversion of the whole text to lower case letters, replacement of the words, removal of several unimportant characters, stop-word removal, and finally, tokenization have been carried out; afterward, the proposed model powered by various computations such as embedding, encoding, attention, and decoding are carried out. The standard metrics of NLP applications such as BLEU, METEOR, and ROUGE have been obtained to evaluate the generated text quality.

The rest of the section of this paper has been organized as follows: background work is presents in section 2, the proposed model is described in section 3, following up we provided results and analysis in section 4, and finally, the conclusion of the paper is presented in section 5.

2. Related Work

In the literature, it can be found that traditional and recent generative models [26, 40] are mostly adopted for text generation. Van der Lee *et al*. [51] have investigated a trainable approach for data-to-text generation influenced by the templatization technique. This approach has relied upon a rule-based method, so it requires manual effort, and this has decreased the text quality based on the BLEU score. Rules-based text generation relies on the language's grammar, syntax, and semantics and it has been used for tasks like text summarization and simplification, but they have limitations as they are confined to predefined rules and may not produce outcomes as natural as human-written text. In the literature, also it can be found that statistical methods such as Hidden Markov Models (HMM) and N-gram have been used for large text corpora for training. This training is accomplished in order to learn patterns and dependencies among input text [19]. Statistical methods have been used in various Natural Language Generation (NLG) tasks, such as language modeling, text summarization, and machine translation. Another different method such as the template-based approach for text generation proposed by Van Deemter *et al*. [50] is notable for its introduction of evolutionary computation techniques, where the authors have included a set of predefined templates and grammar rules. Palivela [30] in his work has shown how to fine-tune a text-to-text transfer transformer model for paraphrase generation. In his work, paraphrasing is achieved in two steps: the first step is known as paraphrase identification, and the second step is paraphrasing generation. Zhao *et al*. [61] have combined BERT and context attention mechanisms to generate short conversations. It combines the standard Seq2Seq model and BERT embedding to improve the quality of the text. BERT and GPT-2 based automatic question and answer generation has been proposed by Kumari and Pushphavati [21]. Their work has focused on generating short answers from the question, where BERT and GPT-2 have been used as encoders and decoders. Barros *et al*. [9] proposed a hybrid surface realization approach named Hana NLG for news headline generation, where the influence of content selection on surface realization is the key focus. They have generated coherent and linguistically structured headlines using Document Understanding Conference (DUC) and DUC 2004 standard datasets. Diwan *et al*. [15] have explored the utilization of AI-based techniques for learning pathway augmentation and content generation and learning to enhance learner engagement. Diwan *et al*. [15] and authors have used GPT-2, where they generate personalized and interactive learning narratives that cater to individual learners. Ayana *et al*. [4] have developed reinforcement learning powered by two models for creating news headlines. Their experimental results outperform headline generation in Chinese-English cross-lingual, but the models are unsuitable for training in different source languages. Wei and Zhang [56] worked on an LSTM-based attention mechanism to generate natural answers that have focused on real-world Knowledge Base (KB) question answering. Wang et al. proposed an interesting work that has generated the introductory parts of any research papers automatically using RNN and text rank algorithms [55]. This work suffers from an out-of-vocabulary problem, which produces meaningless sentences sometimes. Dethlefs *et al*. [12] have developed a divide-and-conquer based on LSTM to generate input context text. However, the authors mentioned that there is a scope for better hyper-parameter tuning in memory to sequence and sequence-to-sequence models. Cao [10] has proposed table-to-text generation from weather gov, Wiki-Bio, and Wiki table datasets by using RNN and LSTM models. The following models are available in the literature mostly for generating semantic-rich text [2, 6, 17, 32, 38, 39, 48]. Chen *et al*. [11] have fine-tuned the BERT model to develop the detection of automatic generation of essays in Chinese language. They built an essay generator through the GPT-2 model and an essay detector by the pre-trained BERT model. Semantically, text generation is possible with either a Large Language Model (LLM) or a hybrid of the sequential model. In the following Table 1, represents the key features and approach for different type of text generation system.

Table 1. Related work in the field of text generation.

Author(s)	Approach	Key Features
Van der Lee *et al.* [51]	Trainable approach for data-to-text generation influenced by templatization.	Rule-based method, manual effort, BLEU score decrease.
Van Deemter *et al.* [50]	Template-based approach for text generation.	Evolutionary computation techniques, predefined templates and grammar rules.
Palivela [30]	Fine-tune text-to-text transfer transformer model for paraphrase generation.	Paraphrase identification and generation.
Zhao *et al.* [61]	Combined BERT and context attention mechanisms for short conversation generation.	Seq2seq model, BERT embedding.
Kumari and Pushphavati [21]	BERT and GPT-2 based automatic question and answer generation.	Short answer generation, BERT and GPT-2 as encoders and decoders.
Barros *et* al. [9]	Hybrid surface realization approach (Hana NLG) for news headline generation.	Content selection influence on surface realization, DUC 2004 datasets.
Diwan *et al.* [15]	AI-based techniques for learning pathway augmentation and content generation using GPT-2.	Personalized and interactive learning narratives.
Ayana *et al.* [4]	Reinforcement learning for news headline generation	Outperforms Chinese-English cross-lingual headline generation.
Wei and Zhang [56]	LSTM-based attention mechanism for natural answer generation.	Real-world KB question answering.
Wang *et al.* [55]	Automatic paper writing using RNN and TextRank algorithms.	Introductory part generation of research papers, out-of-vocabulary problem.
Dethlefs *et al.* [12]	Divide-and-conquer approach based on LSTM for input context text generation.	Better hyper-parameter tuning in memory-to-sequence and sequence-to-sequence models.
Cao [10]	Table-to-text generation using RNN and LSTM models.	Weather gov, Wiki-Bio, and Wiki table datasets.
Chen *et al.* [11]	Fine-tuned BERT for detecting automatically generated essays in Chinese.	Essay generator through GPT-2, essay detector by pre-trained BERT model.

2.1. Text Generation Using Seq2Seq Model

Seq2Seq model was first introduced [46] where the Sutskever *et al*. [39] proposed a single-layer LSTM-based encoder-decoder architecture to translate English to French text. Since then, the Seq2Seq model for text generation can be represented as a series of computations performed on the input and output text. The mathematical representation of the encoder-decoder model is as follows:

$$Embedding\ layer: x_t = Embedding(x_t) \tag{1}$$

$$Encoder\ LSTM/RNN: h_t = Encoder(x_t, h_{t-1}) \tag{2}$$

Equations (1) and (2) represent the text encoder of the model, where x_t and h_t represents the input text, and hidden state at time step t. Embedding and Encoder are functions that are applied to the input text. The embedding function converts the input text dataset into a dense representation and hidden state update by encoder function.

$$Embedding\ layer: y_t = Embedding(y_t) \tag{3}$$

$$Attention\ Mechanism: context = Attention(h_t, h_{t-1}) \tag{4}$$

$$Decoder\ \frac{LSTM}{RNN}: y_t, h_t = Decoder(y_t, h_{t-1}, context) \tag{5}$$

Equations (3), (4), and (5) represent the decoder parts of the sequence model. Where y_t, and h_t are the output text and hidden state respectively at the t time step. Three functions, embedding, attention, and decoder, are applied to the output text. The embedding function i.e., Equation (3) converts the output text into a dense representation, whereas the attention function Equation (4) computes the attention mechanism, and the decoder function generates the output text and updates the hidden state [5].

Figure 2 shows the detailed architecture of encoder-decoder for sentence generation. Here, RNN is the encoder and LSTM is the decoder [55].

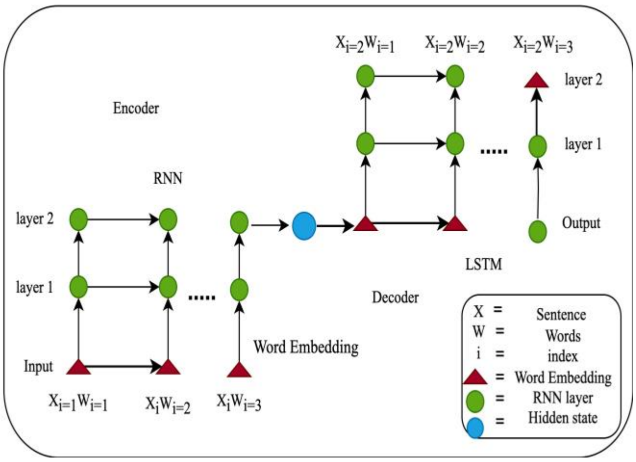


Figure 2. Sentence generation architecture using an LSTM-based encoder and decoder model.

2.2. Contextual Generation Using Bi-LSTM Attention Model

Bidirectional Long Short-Term Memory (Bi-LSTM) consists of two LSTM models. It works in both directions: forward and backward [2, 35]. Bi-LSTM captures contextual information, so it understands the whole sentence context and it generates semantically enhanced text. This typical forward and backward structure is useful for generating coherent and semantically enhanced text [36, 48]. For example, the input text is "john m. walker is" and our model completes this sentence by generating "john m. walker is" an American politician and business man".

Figure 3 shows the overall flow of the text generation using our proposed model. It can be seen from the figure that we first collected the Wiki-Bio dataset which has been represented as W, and n represents the input words. Bi-LSTM captures features such as word interdependency, grammar patterns, co-referencing, and Named Entity Recognition (NER) from the dataset [32]. Next, we have provided initial 2-3 words for text generation as input, represented as outline text which can be seen in Figure 3. Next LSTM with attention model analyzes the outline text and predicts the next word t using the Bi-LSTM feature map. The target text is represented as (M) [37]. Onwards LSTM processes the text features and generates the sentence based on the input corpus.

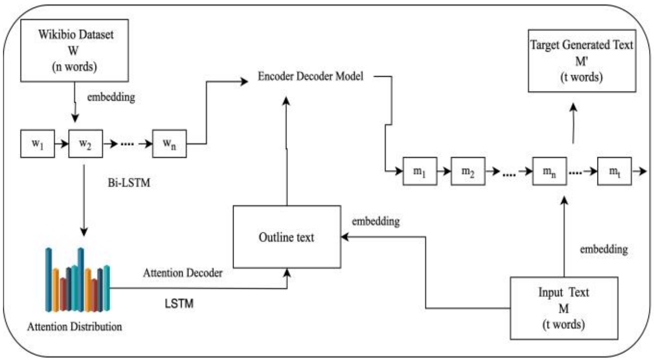


Figure 3. A framework for contextual text generation using LSTM, Bi-LSTM with Attention mechanism on the Wiki-Bio dataset.

In this work, we assigned the embedding layer dimension as 16 and two Bi-LSTM layers with 256 and 512 hidden neurons. We provided the input initial text from Wiki-Bio data "Jon tester born" and it assigns to k k_n^qas the input gate. The a_n is the variable that takes "Jon", "tester", and "born" words as input in the time stamp n=1,2 and 3, respectively.

$$k_n^q = \sigma(w_k, a_n + S_k m_{n-1}^q + o_k^q) \quad (6)$$

$$q_n^q = \sigma(w_q, a_n + S_q m_{n-1}^q + o_q^q) \quad (7)$$

$$i_t^q = \sigma(w_i, a_n + S_i h_{n-1}^q + o_i^q) \quad (8)$$

In Equations (6), (7), and (8), $a_n, m_n^q, m_n^o, p_n^q,\ and\ p_n^o$, represent the forward hidden state, backward hidden state memory cell, backward memory cell, and output get respectively. We have initialized the forward LSTM hidden state m_n^q with zero vector. The forward input gate represents as k_n^qin Equation (6). Next, the variable q_n^q represents the forward forget gate in Equation (7) and i_t^q has been represented as a backward output gate in Equation (8). Further, we assign our initial values "Jon tester born" in Equation (6), which is shown in Equations (9), (10), and (11).

$$k_1^q(1) = \sigma("Jon" + S_0(0)) \quad (9)$$

$$k_2^q(2) = \sigma("tester" + S_1(1)) \quad (10)$$

$$k_3^q(3) = \sigma("born" + S_2(2)) \quad (11)$$

Similarly, in Equations (9), (10), and (11) represents the backward input gate, forget gate, and output gate are represented at time step n as following equations.

$$k_n^o = \sigma(w_k, a_n + S_k m_{n+1}^o + o_k^o) \quad (12)$$

$$q_n^o = \sigma(w_q, a_n + S_q m_{n+1}^o + o_q^o) \quad (13)$$

$$i_t^b = \sigma(w_i, a_n + S_i m_{n+1}^o + o_i^o) \quad (14)$$

In the above Equations (12), (13), and (14), σ represents the sigmoid function, S_n, S_q, S_i represent the weight matrices and o_k^o, o_q^o, o_i^o represents the bias. Further, we assign our initial value in the backward input gate in the below equations

$$k_1^o(1) = \sigma("born" + S_0(0)) \quad (15)$$

$$k_2^o(2) = \sigma("tester" + S_1(1)) \quad (16)$$

$$k_3^o(3) = \sigma("Jon" + S_2(2)) \quad (17)$$

In Equation (15), model assigned a_0="born"; in Equation (16) a_1="tester", and in Equation (17) the variable a_2 has been assigned as "jon". Bi-LSTM learns the sentence structure with the help of a forward hidden state and a backward hidden state. The next step of the process is to decide whether the information is stored in a memory cell or not.

$$p_n^q = q_n^q * p_{n-1}^q\ k_n^q * tanh(W_p a_n + S_p + m_{n-1}^q + o_p^q) \quad (18)$$

$$p_n^o = q_n^o * p_{n+1}^o\ i_t^b * tanh(W_p a_t + S_p + m_{n-1}^o + o_p^o) \quad (19)$$

Equations (18) and (19) are used to calculate forward and backward memory cells at time step n, where, p_n^q is the forward memory cell and p_n^o is the backward memory cell. All initial information are stored in the memory cell, and based on this information output gate predicts the next words by using forward and backward hidden states.

$$m_n^q = i_n^q * tanh(p_n^q) \quad (20)$$

$$m_n^o = i_n^o * tanh(p_n^o) \quad (21)$$

Finally, Equations (20) and (21) represents the forward and backward hidden states concatenate and generate the output at n time step.

$$z_n = g(m_n^q, m_n^o) \quad (22)$$

Equation (22), shows the concatenation of forward and backward hidden state, $[m_n^q, m_n^o]$ and g represents a fully connected layer of linear function. Finally the output z_n represents the processing of the prediction of the next word and generates contextual text.

$$z_{n+1} = "in" \quad (23)$$

Next, the input text is updated with "jon tester born in" in Equation (23). All equations starting from Equations (6) to (23) have generated one word at a one-time step and it has worked in the loop to generate complete words for a sentence. In the following equation, we see that this process has accomplished 3 times and predict the next three words. Next, the input text has been updated with "jon tester born in" in Equation (23) and the Equations (6), (7), and (8), ……up to (22) were again repeated for the next time step t+1, t+2………n till the last word of the sentence is predicted.

$$z_2 = "august" \quad (24)$$

$$z_3 = "21" \quad (25)$$

$$z_4 = "1956" \quad (26)$$

In Equations (24), (25), and (26), our model generates the complete sentence that is *z*="jon tester born in august 21 1956". In this work, the LSTM model has generated the next word in the output sequence, based on the previous sentence context provided by the Bi-LSTM hidden states.

2.3. Data-to-text Generation Using CAN Model

In the convolutional attention model, Convolutional Neural Network (CNN) is used to process the input text by extracting features at different levels of abstraction [62]. In our proposed model, it has extracted the features from the text such as text pattern and grammatical structure of the text; then it has created a word feature vector. Conv1D has analyzed the grammatical ambiguity in the sentence so that error-free sentences can be generated. We used N-gram model for a better understanding of sentence structure. The embedding, Conv1D, and attention functions help the model in the training phase so that it can predict the next word of the sentence.

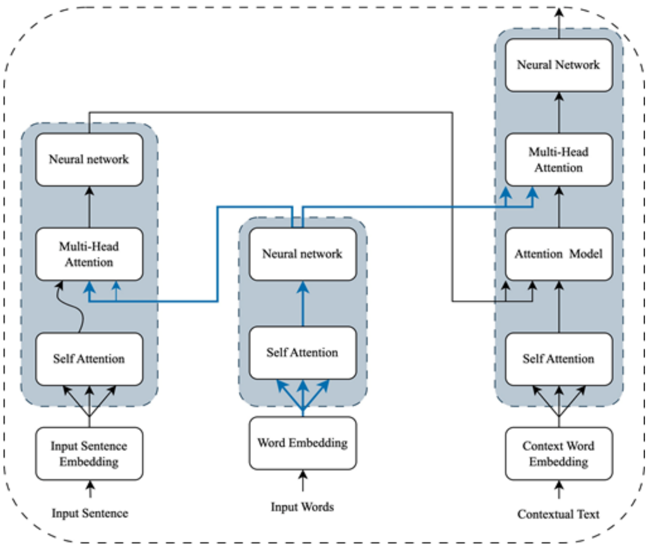


Figure 4. CAN for data-to-text generation.

Figure 4 represents a framework to generate context-based text, here we have used the attention mechanism with Conv1D. Further, this model has enhanced the performances of Bi-LSTM model.

The following equations represent a 1D Convolutional neural network (Conv1D) for a data-to-text generation:

$$h_t = conv1d(x_t, W) + b \tag{27}$$

Equation (27) represents the vector representation of input of the data, of *conv*1*d* which *W*, and *b* represents the convolutional filter and bias at *t* time step. The convolutional layer output is passed through the non-linear activation function ReLU, which is represented below:

$$h_t = f(h_t) \tag{28}$$

In Equation (28), the convolutional layer is passed to a pooling layer, where *f* represents the activation function. Then pooling layer output passed through the fully connected layer, which generates the final output:

$$y_t = g(h_t) \tag{29}$$

In Equation (29), *g* represents a linear function for a fully connected layer, and y_t represents the final output of the model.

2.4. Model Optimization Using Humpback Whale Optimization

The Whale Optimization Algorithm (WOA) has been inspired by the hunting behavior of humpback whales. It has been proposed by Mirjalili and Lewis [27]. The algorithm consists of two main phases: exploration and exploitation. During the exploration phase, the algorithm explores the search space by randomly moving in different directions so that it can achieve the solution or goal state [18]. It allows for broad exploration and helps to discover potential areas of improvement. The exploitation phase usually is used for search so that the best solution can be found quickly in the given search space [25].

WOA utilizes a set of mathematical equations inspired by bubble-net feeding and bubble searching. The following equations govern virtual whales' movement and behavior representing potential solutions in the optimization process [16].

$$\vec{X} = \overrightarrow{2m}.\vec{n} - \vec{m} \tag{30}$$

$$\vec{Y} = 2.\vec{n} \tag{31}$$

$$\vec{Z} = |\vec{Y}.\overrightarrow{T^*}(p) - \vec{T}(p)| \tag{32}$$

$$\vec{T}(p+1) = \overrightarrow{T^*}(P) - \vec{X}.\vec{Z} \tag{33}$$

$$\vec{Z} = |\vec{Y}.\overrightarrow{T_{random}} - \vec{T}| \tag{34}$$

$$\vec{T}(P+1) = \overrightarrow{T_{random}} - \vec{X}.\vec{Z} \tag{35}$$

$$\overrightarrow{Z'} = |\overrightarrow{T^*}(p) - \vec{T}(p)| \tag{36}$$

$$\vec{T}(t+1) = \overrightarrow{Z'}.\overrightarrow{e^{hl}}.cos(2\pi k) + \overrightarrow{T^*}(p) \tag{37}$$

Here $\vec{a}$ represents variable linearly decreases from 2 to 0, with iterations, and $\vec{n}$ represents a random vector in [0, 1]. *h* and $\vec{Z}'$ represent the constant and distance between the whale and best solutions, respectively. *L* shows a random number in [−1, 1]. *P*, $\vec{Y}$, $\vec{T}$, and *T* represent the current iteration, coefficient vectors, and the best solution's position vector and position vector. The behavior of encircling prey has been shown in Equations (30) and (31), Equations (32), and (33) describe the 'search for prey' mechanism, and bubble-net attacking behavior has been shown in Equations (34) and (35), which are exhibited by humpback whales. Additionally, specific modifications have been made, such as hyperparameter tuning with iteration. One notable aspect of WOA, it has easily been implemented with our model to optimize the hyperparameters such as the number of iterations, number of neurons, and number of batch sizes.

2.5. Text Generation Using LLM

LLM is a language model with large number of parameters i.e., the model can be trained with huge amount of data. This model usually are powered by deep learning technique especially such as transformer model [3, 52]. This transformer based LLMs are capable of recognizing, translating, predicting, or generating text. Various transformer models generate text, including popular ones like BERT [13], GPT, and LaMDA [47]. To validate our proposed model, we have compared our proposed model's outcome even with conventional LLM models, such as checking the potential capabilities of BERT and GPT-2 to generate text. In below, we have discussed about BERT model and how it has been used for our use case experimentations.

2.5.1. Bidirectional Encoder Representations from Transformers (BERT)

BERT can convert structured data into readable text using tokenizing techniques. It processes data and remembers contextual connections among tokens for understanding of the context of the text. In our work, we have fine-tuned the BERT-base pre-trained model to predict words or phrases and finally printed the human readable text.

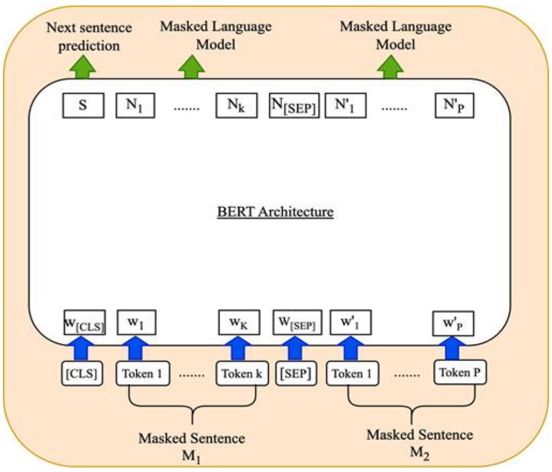


Figure 5. BERT model for data tot text generation.

In Figure 5, every sentence has been separated by [CLS] token, and the word or token is separated with [SEP] token [13]. Here, w represents the word, and in the output layer, S and N represent the sentence and word, respectively. In our work, BERT has employed essential intermediate tokens: [CLS], [SEP], and [MASK]. [CLS] represents the input context, [SEP] marks sentence separation, and [MASK] predicts the next word. Using BERT, we have achieved % accuracy and 0.73, 0.6221, and 0.7612 of BLEU3, ROUGE L, METEOR, and CIDERr score. Our BERT model has generated next-word predictions for a sentence's [mask] label based on surrounding words [13]. Our process uses a BERT tokenizer to convert the sequence into a vector, denoted as $\{y_t\}$ (t=1, 2, 3, ….) where t represents the sequence length. Following this, we proceed to train the BERT model, using the input vector $\{y_t\}$ and saving the model output sequence as $\{output_j\}$ (j=1, 2, 3 ….) where j represents the total number of outputs. To find the next probable text, we use a linear layer,

$$x_i = Softmax(output_j) \tag{38}$$

Equation (38) represents the process $\{output_j\}$ output of BERT, we take the top five $\{y_i\}$ probabilities for the next character in the dictionary for comparison. We run through the model in four datasets, Wiki-Bio, CoNaLa, IMDB and Gigaword where we evaluate the generated text with BLEU, ROUGE, METEOR, and CIDEr.

2.5.2. GPT-2

We have accomplished the successful experimentation using Generative Pre-trained Transformer 2 (GPT-2) powered by the Hugging Face Transformers library in Python. This has helped us immensely as tool for text-generation tasks. In the below Table 2 and 3, it can be observed that how the GPT2 has generated coherent and contextually relevant text across various domains from its BLUE, ROUGH and METEOR and CIDEr score. GPT 2 nowadays is incorporated for creative writing to content generation. In our case, using the Hugging face API integration, we have able to show the power of pre-trained language models like GPT2. The highest BLUE score is 0.75, the maximum score in-terms of ROUGH is, and CIDER metric have the maximum value as 0.6987, 0.6945, and 0.6395. GPT-2 via Hugging Face in Python enhances productivity and fuels innovation in NLP and text generation. GPT-2, which is an upgraded version of GPT-1, excelled in various tasks by predicting sequence items accurately but occasionally produced repetitive or nonsensical content in longer passages. It's been succeeded by non-open-source GPT-3 and GPT-4 [23].

Algorithm 1: For Text Generation Using GPT2.

```
Input     Keyword k, model M, length l
Output    Sentence y
          y ← k
          for j=len(k) to l do
                    M.init()
                    next word ← M(y)
                    next word ← M(y)
                    y ← y + next word
          End for
          return y
```

In the above Algorithm (1), the keyword k represents the starting word, the model M is the GPT2 model, and l represents the length of the output sentence. Sentence y is the final generated sentence, which the model returns. Here, the input form is defined as {k, M, l} [29].

3. CANBLWO for Generation of Semantically Enhanced Text

This section describes the proposed model of Convolutional Attention Bi-LSTM with Whale Optimization (CANBLWO). In our proposed hybrid model, semantic text generation has been achieved through data pre-processing, feature map creation using CAN encoder, text generation using Bi-LSTM decoder, model optimization using WOA, and finally text quality testing with evaluation matrices such as BLEU, METEOR, ROUGE, and CIDEr. These metrics have been used to find the ratio between reference text and machine-generated text and the ratio represents the quality.

3.1. Data Pre-Processing

In our work, we have applied various pre-processing steps such as lowercasing, tokenization using the NLTK Punkt library, and data cleaning using Python libraries like pandas for removing duplicate entries, and regular expressions to correct inconsistent formatting. In the pre-processing, we also define the end of a sentence by <end> symbol, which help us to identify the compilation of the sentence and the start of the next sentence. In the final step, we applied Scikit learn library for the data validation, where it handled missing values and checked data consistency in terms of dates and time formatting.

3.2. Feature Gathering from Input Text

The N-gram model extracts the features from the dataset. This model helps in process and understanding the text [25, 38]. In this work, the bigram model has been used for the Wiki-Bio dataset to identify common two-word phrases such as "United States" or "Prime Minister," while in the CoNaLa dataset, a trigram model has been used to identify common sequences of words in natural language commands such as "show me all" or "give me the".

The trigram model has considered the previous two words as context to predict the next word. By considering this contextual information, it generates a probability distribution for the vocabulary of the dataset. w_i is the word probability with preceding context k_{j-2}, k_{j-1} would be as follow:

$$P(k_j|k_{j-2}, k_{j-1}) = count(k_{j-2}, k_{j-1}, k_j) \;/\; count(k_{j-2}, k_{j-1}) \qquad (39)$$

Equation (39) shows the probability distribution of N-gram trigram model, where *count*(k_{j-2}, k_{j-1}, k_j) is the total number of times of trigram, where (k_{j-2}, k_{j-1}, k_j) appears in the corpus. *count*(k_{j-2}, k_{j-1}) is the total number of times of bigram (k_{j-2}, k_{j-1}) is presented in the dataset.

Figure 6 represents the sample of word prediction by N-gram model, where 3 is the value of n. In every step, three-word sequence appears for the process, and on the basis of these three words next word has predicted.

$$P(k_j|k_{j-2}, k_{j-1}) = count(\, k_{i-1}, k_j) \;/\; count(\, k_{j-1}) \qquad (40)$$

The bigram probability distribution is shown in Equation (40), *count*(k_{j-1}, k_j) is the total number time of bigram of (k_{j-1}, k_j) appears in the corpus.

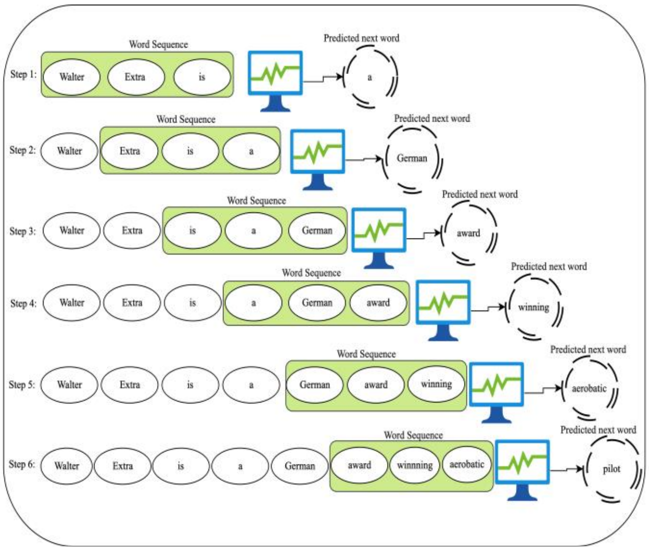


Figure 6. N-gram word prediction with trigram model.

3.3. Data Encoding

We have used the Embedding from Language Model (ELMO) data encoding technique [41]. It has been used for contextual text generation on the Wiki-Bio, CoNaLa, IMDB and Gigawords datasets, which provided a sentence structure and word meaning of the input data. ELMO generates word embedding for the input data that can be used for our hybrid model CANBLWO to generate text. It is a context-dependent embedding method that provides a good understanding of different words in the input data [33], the ELMO model is trained using the following mathematical function:

$$J(\theta) = \sum_i log\, p(w_i|w_1 : i-1, \theta) \qquad (41)$$

In Equation (41), θ is the model parameter, w_i is i^{th} the word for input sequence, and the probability of i^{th} word is $p(w_i|w_1{:}i\text{-}1)$ for input sequence given by the previous word sequences.

3.4. Text Generation Model Architecture

We mentioned earlier that the proposed model is a fusion of CAN, Bi-LSTM, and WOA. Conv1D model extracts the internal features from the text, such as identifying the subject, verb, and object from the sentence. It has found all the Part of Speech (POS) tags and entity relationships in the sentence words. POS tag has helped to understand each word's context, and entity relationship has been used to identify the interdependency relationship of the text in a sentence. For example, we have a sentence from the Wiki-Bio dataset "john m. walker is an American politician and businessman". The proposed CAN first tokenizes the sentence into Part-Of-Speech tags (POS) for every token. In the POS tag report, “john”, “m.”, and “walker” are referred to as consecutive nouns so they can represent as person's full name.

The proposed CAN model also finds the referencing among sentences such as “john m. walker is an American politician and businessman from Colorado. A republican, he is a member of the Colorado state senate”. The attention models divide the sentence into small ones such as “john m. walker is an American politician”, “john m. walker is a businessman”. “john m. walker from Colorado”, and “john m. walker is an American politician member of Colorado state senate”. In all sentences, the subject is mapped with john m. walker. Likewise, the proposed attention model i.e., CAN modes builds the feature maps which further can be fed to the Bi-LSTM.

Next the proposed Bi-LSTM model decodes the feature map and generates the text based on initial input. Bi-LSTM encodes initial text into fixed-length representation, and it also supports long dependency. For example, the initial word “john m. walker is an”, Bi-LSTM considers this initial text as input and encodes this input text into a fixed-length representation. To start the text generation process, a special token <start> is provided as the first input to the decoder. The decoder takes the initial word one by one and the encoded context as input, to generate the next word in the sequence so that prediction of the most likely word can be accomplished based on the current context. The newly generated word is now embedded into a dense vector representation to capture its semantic meaning. The embedded word and the updated context are fed back into the decoder. Decoder utilizes the recurrent connections within the Bi-LSTM to capture the dependencies between the previously generated sentence and the current word context. In every step, the decoder outputs depend on a probability distribution over the vocabulary. This distribution is generated using a softmax activation function. After this generated word is appended to the previously generated words and forms a new partial sentence. This partial sentence, along with the updated context, has been used as input for the next iteration of the decoding process. These steps are repeated until reaching a maximum sentence length or generating a special token <end>. Finally, the complete generated sentence by our proposed model is “John m. walker is an American politician and businessman from Colorado”.

Our proposed model consists of the following three main approaches.

- Capture local features of text using CAN.
- Capture long-term dependency using the Bi-LSTM model.
- Optimize the learning rate, number of layers, and neurons using WOA.

In Equation (42), x represents input data, and it passes through the Conv1D encoder to generate the feature map. The attention mechanism calculates the attention weight, represented by the symbol ‘a’ for each position of the ‘h’ feature map. Here, h and a have been fed into decoder Bi-LSTM that in terms predicts one word at a time. The complete operation is shown below.

- Encoding: x_i is an encoded vector representation of input data x in Equation (42).
- Feature extraction: the encoded input data is passed through a Conv1D encoder to extract features from the data. We have provided different neuron sizes, such as 64, 128, and 256 for feature selection, finally at 256 neuron size provides better results in the model stacking. We have used two layers of Conv1D, and shape of the layer is (5, 32, 128) and (5, 128, 256) respectively.

$$h_t = conv1d(x_i) \tag{42}$$

- Attention: the feature map is passed through an attention mechanism to calculate attention weights for each position in the feature map, which is shown in Equation (43).

$$a = Attention(x_i) \tag{43}$$

- Text generation: while generating text, the feature map and attention weights are passed through Bi-LSTM decoder. Equation (44) shows the Bi-LSTM decoder model,

$$y_t = Bi - LSTM(h, a, y_{t-1}) \tag{44}$$

where y_t represents generated text at time step t. y_{t-1} is the previous time step of generated text.

- Loss function: cross-entropy has been used as a loss function for model training. Basically, it finds the difference between ground truth output and predicted output.

$$L = -sum(y_t * log(y_{h,a,t}) \tag{45}$$

In the Equation (45), loss function L calculates the error between the predicted probabilities $y_{h,a,t}$ and the true labels y_t. The negative summation ensures that higher probabilities assigned to the correct class lead to lower loss values.

Figure 7 represents the contextual text generation model using Conv1D, Attention, and Bi-LSTM. The following layers have been essential in our proposed model.

- Embedding layer: in this layer each data point is converted into a numerical vector representation using an embedding matrix w_e. Shape of embedding layer is (10622, 32), where 10622 is vocab size and the dimension is 32. Mathematical representation of the embedding layer is as follows

$$x_i = w_e[i] \tag{46}$$

In Equation (46), i represents the index of words in the vocabulary. Here, the input data is passed through a series of one-dimensional convolutional filters to extract local patterns in the biography data from Wiki-Bio. The mathematical representation of the layer is

$$x_i = y_i = conv1d(x_i, w_c) + b_c \quad (47)$$

where w_c, b_c are the filter weights and bias terms, respectively in Equation (47).

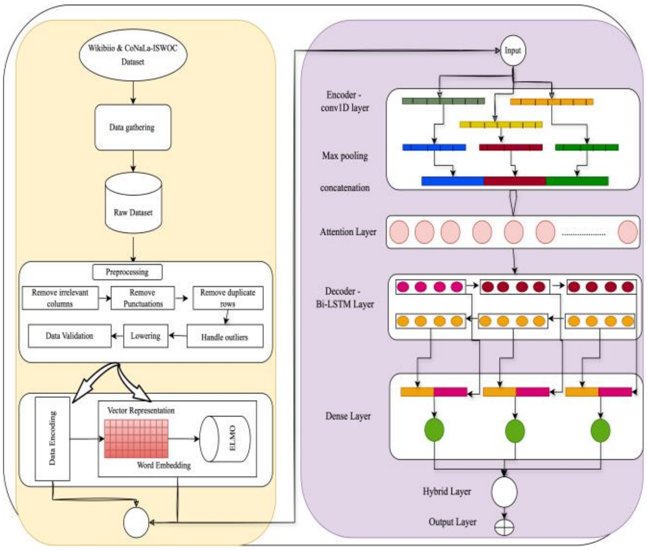


Figure 7. CAN with Bi-LSTM model architecture to contextual text generation.

- Attention layer: this layer selectively focuses on certain parts of input data. It uses the dot product of a set of learned weights and input data. The representation of the attention layer in terms of the equation can be written as follows:

$$a_i = softmax(y_i * w_a) \quad (48)$$

In the above Equation (48), we proposed the weight modification which represents by w_b

$$a_i = softmax((y_i * w_a) + w_b) \quad (49)$$

$$z_i = sum(a_i * y_i) \quad (50)$$

Equations (49) and (50) calculate the attention weight, where attention weight is w_a, and attention output is z_i.

- Bi-LSTM layer: feature map of the attention layer z_i is the input for this layer. It processes in both directions to capture long-term dependencies in paragraphs or sentences of Wiki-Bio dataset.

$$h_i = BiLSTM(z_i, h_{i-1}) \quad (51)$$

In Equation (51), h_i is represents hidden state at t time step.

- Dense layer: h_i represents the output of the Bi-LSTM layer which contains all sentence dependencies such as pronoun-to-noun mapping and the context of the initial text. That dependency and context are input for the dense layer, which projects onto vocabulary space. This layer generates a probability distribution by using the SoftMax activation function. It predicts the next word in the sequence over the vocabulary.

$$p_i = softmax(w_d * h_i + b_d) \quad (52)$$

Equation (52) is the mathematical representation of dense layer, where w_d represents dense layer and bias term denoted as b_d.

Algorithm 2: Algorithm for CAN based Bi- LSTM.

Input	*Wiki Bio, CoNaLa, IMDB and Gigaword dataset*
Output	*Semantic rich text generation based on dataset*
Step 1.	*Begin*
Step 2.	*Import dataset*
Step 3.	*Data cleaning: Removing irrelevant columns, handling missing values, Removing duplicate rows, Data validation*
Step 4.	*Data tokenization*
Step 5.	*Convert into bi-gram and trigram sequence of token*
Step 6.	*Token pre-padding as per the max length of sentence*
Step 7.	*Model Building*
	Step 7.1. Embedding Layer with dimension 16: $x_i = w_e[i]$
	Step 7.2. Conv1D layer: $y_i = conv1d(x_i, w_c) + b_c$
	Step 7.3. Attention layer: $a_i = softmax(y_i * w_a)$ & $z_i = sum(a_i * y_i)$
	Step 7.4. Bi-LSTM layer: $h_i = Bi-LSTM(z_i, h_{i-1})$
	Step 7.5. Dense layer: $p_i = softmax(w_d * h_i + b_d)$
Step 8.	*Fitting of the network means adapting the weight on a training dataset*
Step 9.	*Evaluation the network*
Step 10.	*Make prediction of next word and generate the whole text*
Step 11.	*End*

Algorithm (2) represents the stepwise process of generating the text using the CAN with Bi-LSTM model.

3.5. Model Hybridization with WOA

WOA has inspired by the hunting behavior of humpback whales. It combines exploration and exploitation strategies to search for optimal solutions in a search space [28, 45]. This algorithm has been used in our proposed model to fine-tune hyper-parameters such as number of neurons, learning rate and adding the epoch size of the model.

Figure 8 represents the architecture of the proposed model CANBLWO. It represents the stepwise process of model architecture for generating semantically enhanced text. In Figure 8, whale optimization uses an objective function to evaluate the model's performance based on text generation quality metrics. The WOA algorithm updates the positions of the whales iteratively by considering the current best solution, random values, and coefficients. To optimize the filter values of the convolutional layers, it searches space and exploits promising regions. The process continues for a predefined number of iterations, allowing the algorithm to converge toward optimal or near-optimal solutions.

Semantic text generation using the CANBLOW model has been defined as

$$Let\ X = (x_1, \ldots, x_n) \tag{53}$$

$$Let\ Y = (y_1, \ldots, y_m) \tag{54}$$

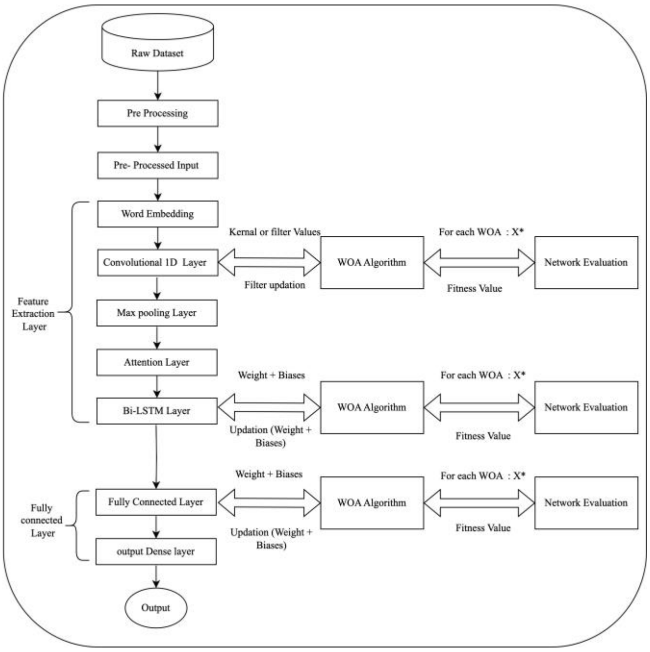


Figure 8. Proposed model architecture powered by CANBLWO.

In Equation (53), X is the input sequence, where x_i represents the *i-th* input token. In Equation (54), Y is the generated output sequence, where y_i represents the *i-th*.

$$E(X) = (e_1, \ldots, e_n),:\ where\ e_i \in \mathbb{R}^d \tag{55}$$

Equation (55) represents the embedding layer, where E is an embedding function, e_i is an embedded representation of the input token x_i, and d represents the embedding dimension.

$$h = \varphi(W\_c * E(X) + b\_c) \tag{56}$$

Equation (56) represents the CAN, φ represents activation function (e.g., ReLU), W_c is the convolutional weight matrix, b_c is the convolutional bias vector and h shown feature map output from convolution.

$$\alpha = softmax(W\_a\, h + b\_a) \tag{57}$$

In Equation (57), W_a, and b_a represent the attention weight matrix and attention bias vector, respectively. α represents the attention weights.

$$z = \Sigma_i\, \alpha_i h_i \tag{58}$$

Equation (58) represents the context vector z, which aggregates information from the input weighted by the attention context vector.

$$\rightarrow h\_t = LSTM(z\_t, \rightarrow h\_\{t-1\}) \tag{59}$$

$$\leftarrow h\_t = LSTM(z\_t, \leftarrow h\_\{t+1\}) \tag{60}$$

Equations (59) and (60) represent the Bi-LSTM layer, which processes sequential data and captures long-term dependencies. In Equation (59) $\rightarrow h\_t$ represents the forward hidden state at time t, and $\leftarrow h\_t$ in Equation (60) is the backward hidden state at time t. z_t represents the context vector at time t.

$$h\_t = [\rightarrow h\_t; \leftarrow h\_t] \tag{61}$$

In Equation (61), h_t represents the concatenated bidirectional hidden state at time t, where [;] is for concatenation operation.

$$p(y\_t | y < t, X) = softmax(W\_o\, h\_t + b\_o) \tag{62}$$

Equation (62) represents the output layer to produce output probabilities, Where $p(y\_t|y<t, X)$ is a probability distribution over vocabulary for the next token. W_o and b_o represents the output weight matrix and output bias vector

$$y\_t = argmax\, p(y\_t | y < t, X) \tag{63}$$

In Equation (63) y_t is generated token at time t and argmax is a function that returns the element with the highest probability.

$$L(\theta) = -\Sigma^t \log p(y\_t | y < t, X) \tag{64}$$

In Equation (64), $L(\theta)$ is Loss function and θ represents all learnable parameters of the model.

$$\theta(t+1)=\{\theta^*-A|C\,\theta^*-\theta(t)|\ \ if\, r<0.5\ D'e^\{b\ell\}cos(2\pi\ell)+\theta^*\ \ else\ \theta(t): Model\ parameters\ at\ iteration\ t \tag{65}$$

Equation (65) represents the best-performing set of parameters found so far by the WOA. Here θ^* is the current best solution. A and C represent the coefficient vectors that control the search behavior in the WOA. D' is the distance to the best solution, b represents the constant for defining spiral shape, ℓ is the random number in [-1, 1], and r denotes the random number in [0, 1].

$$\theta * = argmin\_\theta\ L(\theta) \tag{66}$$

In Equation (66), θ^* represents optimal model parameters.

The complete CANBLWO model for text generation can be expressed as:

$$F_{\theta(X)} = Y = (y^1, \ldots, y_m), where\ y_t = argmax\, p(y\_t | y < t, X) \tag{67}$$

Equation (67) represents the entire text generation process where F_θ denotes the complete CANBLWO model function.

$$p(y_t | y < t, X) = softmax(W\_o\, BiLSTM(CAN(E(X))) + b\_o) \tag{68}$$

In Equation (68), Bi-LSTM denotes bidirectional-LSTM function and CAN represents convolutional attention network function

The CANBLWO model processes input X through an embedding layer $E(X)$, followed by a Convolutional Attention Network (CAN) that extracts features h and computes attention weights α to produce a context vector z. This context is then processed by a Bi-LSTM to capture temporal dependencies, resulting in hidden states *h_t*. The model generates text *Y* by iteratively predicting the next token *y_t* based on the probability distribution *p*(*y_t*|*y*<*t*, *X*). The model's parameters *θ* are optimized using the WOA to minimize the loss function *L*(*θ*).

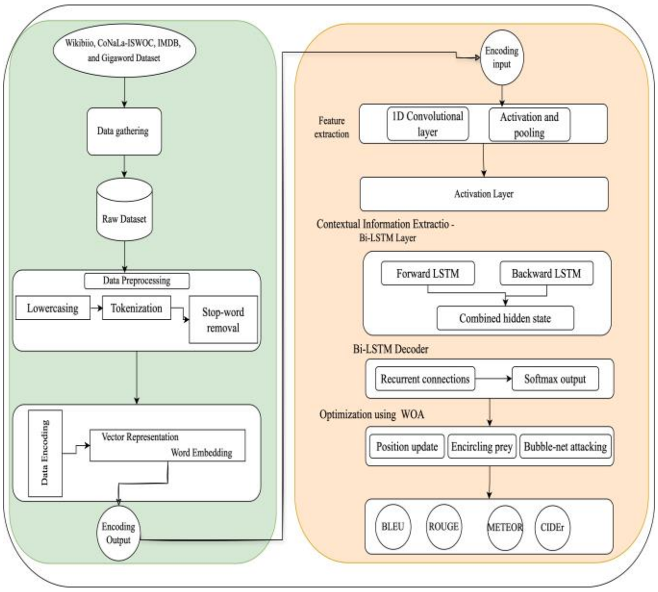


Figure 9. Flow diagram of proposed model CANBLOW.

Figure 9 represents the flow diagram of proposed model CANBLOW, Where the proposed CANBLWO system for semantic text generation utilizes datasets from Wiki-Bio, Code/Natural Language-Intent Shift With Open Code (CoNaLa-ISWOC), IMDB, and Gigaword. The data preprocessing phase includes lowercasing, tokenization, and stop-word removal, followed by encoding into vector representations and word embeddings. Feature extraction is performed using a 1D convolutional layer, which applies filters to extract relevant features, enhanced through activation functions and pooling operations. These features are aggregated in an activation layer for further processing.

Contextual information is extracted using a Bi-LSTM layer, combining forward and backward LSTM units to capture comprehensive contextual information. A Bi-LSTM decoder with recurrent connections generates the next word in the sequence using a softmax function. Model optimization is achieved using the WOA, involving position updates, encircling behavior, and bubble-net attacking strategies. The system evaluates the generated text with BLEU, ROUGE, METEOR, and CIDEr metrics, ensuring high-quality semantic text generation.

The following algorithm is used to hybridize of CANBLWO model.

Algorithm (3) represents the optimization technique, which is used for hybridization of the CANBLWO model. This algorithm firstly initializes whale population T_i (*i*=1, 2, 3……*n*) within the search space and the objective function that is mentioned in the algorithm is used to evaluate the model performance. Step 5.1.1. is used to update the value of A, C, and D which refer to Equations (30), (31), and (32). $\vec{T}(p+1)$ and $\overrightarrow{T^*}(p)$ are the current positions of the whale and the best position respectively. The optimization positions are referred to in Equations (35) and (37). WOA utilizes the positions of the best and current solutions as guidance to search techniques.

Algorithm 3: Algorithm for Optimization of CANBLWO.

Step 1. *Begin*
Step 2. *Initialization of the whale population* $T_i(i = 1,2,3 \dots \dots n)$
Step 3. *Calculation of the objective function of each whale based on model performance*
Step 4. $T^* = Best\ objective\ function$
Step 5. *Loop: while (i < maximum number of iteration):*
Step 5.1. *for each whale*
Step 5.1.1. *Update X, Y, and Z*
$$\vec{X} = \overrightarrow{2m}.\vec{n} - \vec{m}$$
$$\vec{Y} = 2.\vec{n}$$
$$\vec{Z} = abs\ (|\vec{Y}.\overrightarrow{T^*}(p) - \vec{Y}(p)|\ )$$
Step 5.1.2. *Calculate the new position of* $\overrightarrow{T^*}(p)$
$$\vec{T}(p+1) = \overrightarrow{T^*}(p) - \vec{X}.\vec{Z}$$
Step 5.1.3. *If (k < 0.5) and (abs(X) < 1):*
Step 5.1.3.1. *Updating the current position search agent or objective function*
$$\vec{T}(p+1) = \overrightarrow{Z'}.\overrightarrow{e^{hk}}.\cos(2\pi k) + \overrightarrow{T^*}(p)$$
Step 5.1.4. *else*
Step 5.1.4.1. $\vec{T}(p+1) = \overrightarrow{Z'}.\overrightarrow{e^{hk}}.\sin(2\pi k) + \overrightarrow{T^*}(p)$
Step 5.2. *end for loop*
Step 5.3. *If any search goes beyond the search space end it*
Step 5.4. *Process optimization based on evaluation matrix*
Step 6. *End while*
Step7. *Return X **

3.6. Experimental Settings

The experimental setup utilizes 8 GB of RAM, Intel i5 processor, and macOS as an operating system. In addition, the experimentation of the proposed CANBLWO model has been carried out on Google Collab Pro hosted Jupiter notebook. Tesla V100-SXM2-16G GPU and 12.7 GB of RAM have been set as the configuration of the collab notebook. We have used the Python library Keras [20] and Google TensorFlow library [1] as open-source software for our experiments. The details of the software and hardware configuration are shown in Table 2.

Table 2. Experimental configuration.

Experimental configuration	
Processors	Intel i5
Operating system	macOS
RAM	8Gb
Google colab pro GPU	Tesla V100-SXM2-16GB
Google colab pro RAM	12.7GB
Tensorflow Version	2.12.0

4. Result and Discussion

In below, we have discussed detailed results that we have obtained by our proposed model. As mentioned earlier our proposed model is a combination of CAN, Bi-LSTM, and WOA. Prior to that we also discussed the mechanism of preprocessing which included tokenization, stopword removal, lower casing and word sequence generation on the data sets (Wiki-Bio and CoNaLa dataset) and other necessary steps required for building the input corpus.

4.1. Dataset

We performed the experimentation on two well-known standard datasets namely Wiki-Bio, CoNaLa, IMDB and Gigaword. Wiki-Bio dataset has been collected from Wikipedia and it is frequently used in NLP research such as biography generation, table-to-text generation [8, 10, 43], and the CoNaLa dataset has been obtained from stack overflow [59].

Figure 10. Word cloud graph using Wiki-Bio dataset.

The Wiki-Bio data set is a large dataset that comprises of articles of biography that used to get stored in info-box tables [22]. Infobox tables are common features of Wikipedia articles, and they provide key information about the subject of the article, such as date of birth, occupation, and education etc. It contains over 358,321 Wikipedia articles. The articles are sourced from English Wikipedia, which cover extensive subjects, including people, organizations, and locations. The infobox table usually are represented in a structured format, such as JSON or XML. It has been designed to train machine learning models to generate natural language text. In our work, we have trained our proposed model namely CANBLWO to understand the structure of the infobox tables and the relationships between the different pieces of information. The Wiki-Bio dataset has been used in several research studies [8], and it has been proven to be a valuable resource for text generation. It is also a benchmark for the performance evaluation of data-to-text generation models. Figure 10 represents the word cloud of the Wiki-Bio dataset, where the larger font words represent the significant words in the Wiki-Bio dataset [49].

We also have considered CoNaLa dataset which comprises a large collection of question-query pairs from stack overflow. Researchers at the university of California, and Maryland, created this dataset [54]. CoNaLa dataset mainly comprise of programming and sciences related data. It has 2379 number of training examples and 500 test examples. Every example has natural language related information and its corresponding answer. In our proposed work, the CANBLWO model has been able to generate the various questions automatically by feeding input as two or three words.

The IMDb Reviews dataset is a widely used benchmark for sentiment analysis, NLP and NLG tasks. It consists of 50,000 highly polar movie reviews labeled as either positive or negative, split equally into training and testing sets. This dataset allows researchers to evaluate and develop models for sentiment classification, making it a critical resource for understanding how well algorithms can interpret and analyze subjective text.

The Gigaword dataset, on the other hand, is a comprehensive and large-scale collection of news articles from various sources, including the New York Times, Associated Press, and the Washington Post. This dataset comprises over 4 million news articles, providing a rich resource for tasks such as text summarization, language modeling, and text generation. The diversity and volume of the Gigaword corpus make it an invaluable asset for training and evaluating NLP models, especially for developing algorithms that can handle a wide range of topics and writing styles.

4.2. Result and Analysis

This section compared the performance of our proposed model CANBLWO with other deep neural network models such as RNN, Bi-LSTM, LSTM, and Bi-LSTM with the attention mechanism on Wiki-Bio and CaNoLa datasets. The outcome of this comparative study has been shown in Table 3 in terms of accuracy and loss. Table 3 shows a comparative study of various deep learning models' performance on Wiki-Bio, CoNaLa, IMDB, and Gigaword datasets. For Wiki-Bio data set our proposed model CANBLWO has achieved 97% accuracy, followed by Bi-LSTM (87%), RNN (85%), and others. On the other hand, in terms of loss, CANBLWO has the lowest (0.6109), followed by Bi-LSTM with Attention (0.9744), RNN (1.089), and others. In addition, we also have conducted experiment on CoNaLa data set. It can be seen that CANBLWO again excels with 96% accuracy and 0.7469 loss, while Bi-LSTM with Attention follows with 90% accuracy. Our proposed CANBLWO model achieves 97% accuracy on Wiki-Bio and 96% accuracy on the CoNaLa dataset. Additionally, our experiments on the IMDb and Gigaword datasets further demonstrate the robustness of CANBLWO. The model consistently outperforms other approaches, underscoring its effectiveness across different types of text generation task. The accuracy graph is shown in Figures 10.

Table 3. Accuracy and loss comparison on the Wiki-Bio and CoNaLa, IMDB and Gigaword dataset.

Dataset	Model	Accuracy	Loss
Wiki-Bio	RNN	85%	1.089
	LSTM	81%	1.1950
	Bi-LSTM	87%	0.8438
	Bi-LSTM with attention	84%	0.9744
	BERT	96%	0.4234
	GPT 2	92%	0.84
	CANBLWO	97%	0.6109
CoNaLa	RNN	90%	0.8726
	LSTM	71%	1.6195
	Bi-LSTM	89%	1.3178
	Bi-LSTM with attention	90%	1.2565
	BERT	94%	0.6254
	GPT 2	93%	0.90
	CANBLWO	96%	0.7469
IMDB	RNN	72%	1.124
	LSTM	75%	1.041
	Bi-LSTM	78%	0.936
	Bi-LSTM with attention	80%	0.821
	BERT	88%	0.512
	GPT 2	85%	0.672
	CANBLWO	90%	0.489
Gigaword	RNN	68%	1.251
	LSTM	70%	1.164
	Bi-LSTM	73%	1.043
	Bi-LSTM with attention	75%	0.928
	BERT	84%	0.693
	GPT 2	81%	0.821
	CANBLWO	87%	0.632

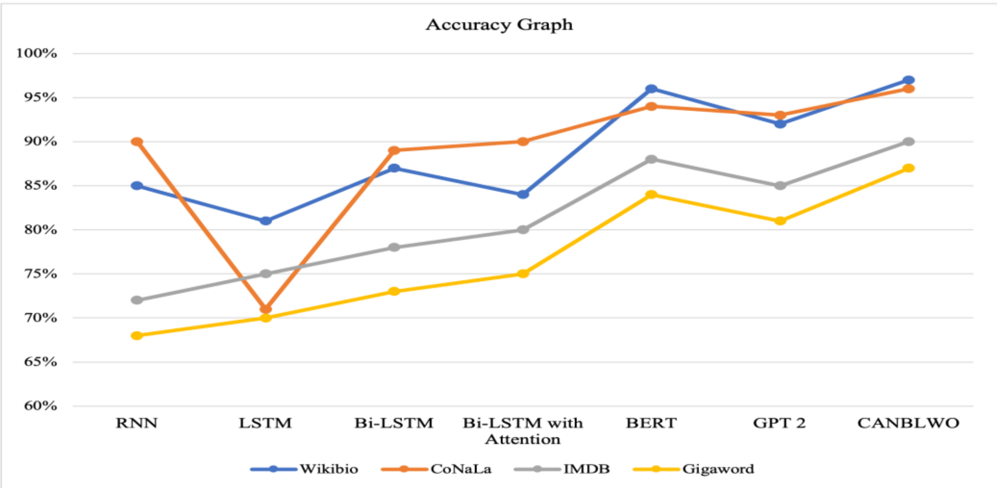


Figure 11. Accuracy graph on Wiki-Bio, CoNaLa, IMDB and Gigaword.

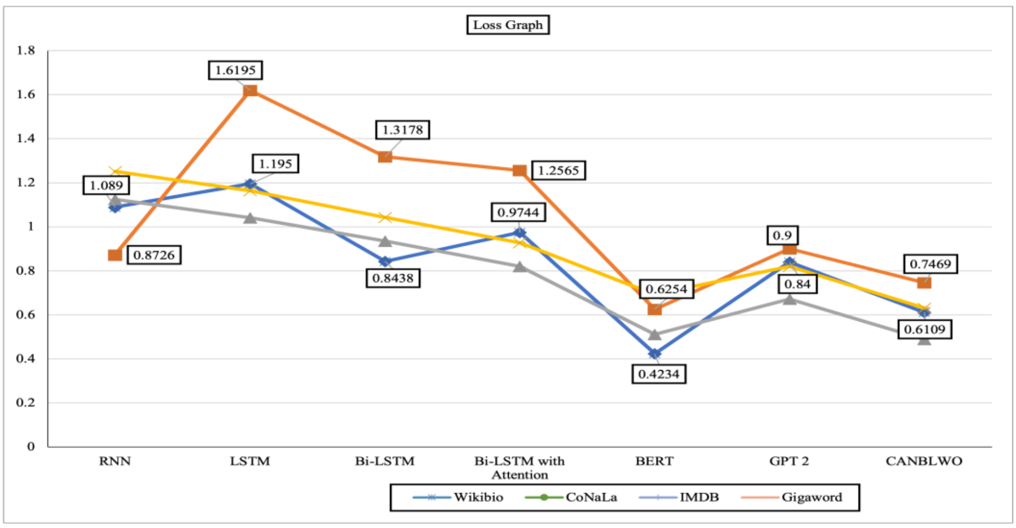
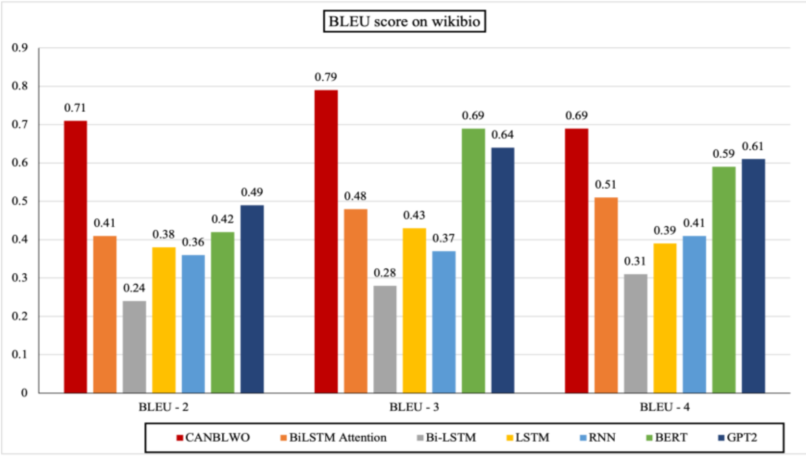
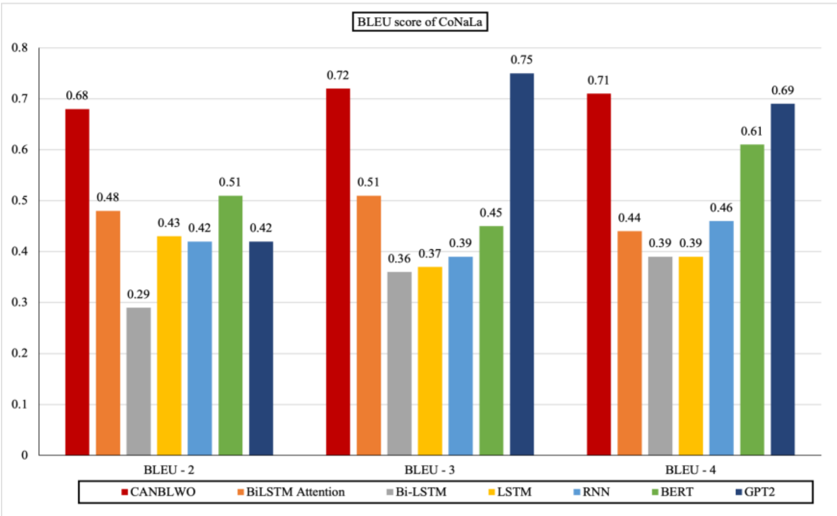


Figure 12. Loss graph on Wiki-Bio, CoNaLa, IMDB and Gigaword.

The accuracy of Wiki-Bio and CoNaLa datasets present in Figure 11. Figure 12 represent the loss graph. We also have compared the generated text evaluation score using different popular metrics such as BLEU, ROUGE, CIDEr, and METEOR evaluation matrices. The below figure shows the comparison graph of these metrics.

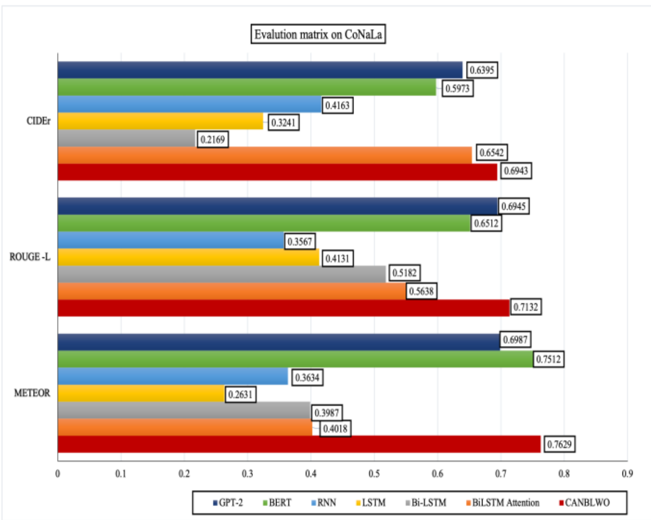


a) BLEU score on Wiki-Bio.

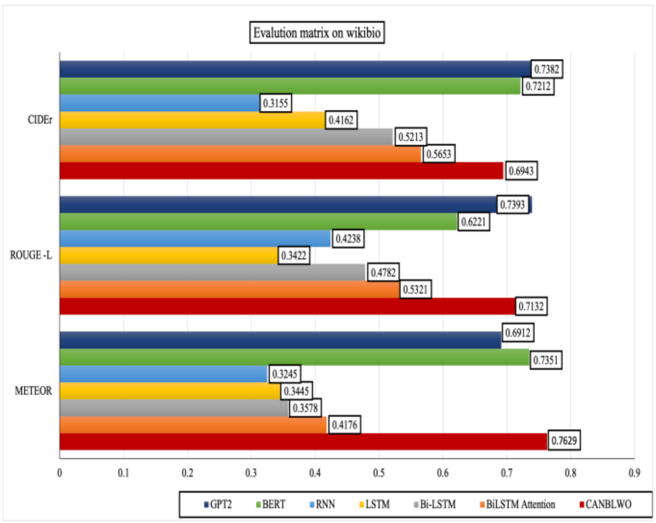


b) BLEU score on CoNaLa.

Figure 13. BLEU-2, BLEU-3, and BLEU -4 score comparison on Wiki-Bio and CoNaLa.



a) Evaluation metrics score on CoNaLa.



b) Evaluation metrics score on Wiki-Bio.

Figure 14. METEOR, CIDEr, and ROUGE score comparison on Wiki-Bio and CoNaLa.

Figure 13-a) and (b) represent the BLEU-2, BLEU-3, and BLEU 4 scores on Wiki-Bio and CoNaLa datasets. Figure14-a) and (b) show a comparison of different evaluation metrics such as ROUGE,

METEOR, and CIDEr score using Wiki-Bio and CoNaLa dataset.

4.3. Evaluation Matrices

To evaluate the generated text, we have used BLEU, METEOR, ROUGE, and CIDEr scores has been used to evaluate the quality of the machine-generated text using our proposed model.

- Bi-Lingual Evaluation Understudy (BLEU): the BLEU mechanism has been used to evaluate the quality of the generated text [31]. It is based on the concept of N-gram, where we compare word by word of generated text and reference text. It is calculated as a precision score by N-gram geometric mean. The score ranges from 0 to 1, where 1 is a perfect match to the reference text. The following equation has been used to calculate the BLEU score.

Table 4. Performance comparison on the Wiki-Bio, CoNaLA, IMDB and Gigaword dataset.

Dataset	Model	BLEU -2	BLEU -3	BLEU -4	METEOR	ROUGE_L	CIDEr
Wiki-Bio	RNN	0.36	0.37	0.41	0.3245	0.4238	0.3155
	LSTM	0.38	0.43	0.39	0.3445	0.3422	0.4162
	Bi-LSTM	0.24	0.28	0.31	0.3578	0.4782	0.5213
	Bi-LSTM with attention	0.41	0.48	0.51	0.4176	0.5321	0.5653
	BERT	0.42	0.69	0.59	0.7351	0.6221	0.7212
	GPT 2	0.49	0.84	0.61	0.6912	0.7393	0.7382
	CANBLWO	0.71	0.64	0.69	0.7629	0.7132	0.6943
CoNaLa	RNN	0.42	0.39	0.46	0.3634	0.3567	0.4163
	LSTM	0.43	0.37	0.39	0.2631	0.4131	0.3241
	Bi-LSTM	0.29	0.36	0.39	0.3987	0.5182	1.2169
	Bi-LSTM with attention	0.48	0.51	0.44	0.4018	0.5638	0.6542
	BERT	0.51	0.45	0.61	0.7512	0.6512	0.5973
	GPT 2	0.42	0.75	0.69	0.6987	0.6945	0.6395
	CANBLWO	0.68	0.72	0.71	0.7629	0.7132	0.6943
IMDB	RNN	0.36	0.39	0.41	0.3245	0.4238	0.3155
	LSTM	0.38	0.43	0.39	0.3445	0.3422	0.4162
	Bi-LSTM	0.24	0.28	0.31	0.3578	0.4782	0.5213
	Bi-LSTM with attention	0.41	0.48	0.51	0.4176	0.5321	0.5653
	BERT	0.42	0.69	0.59	0.7351	0.6221	0.7212
	GPT 2	0.49	0.64	0.61	0.6912	0.7393	0.7382
	CANBLWO	0.71	0.79	0.69	0.7629	0.7132	0.6943
Gigaword	RNN	0.42	0.39	0.46	0.3634	0.3567	0.4163
	LSTM	0.43	0.37	0.39	0.2631	0.4131	0.3241
	Bi-LSTM	0.29	0.36	0.39	0.3987	0.5182	1.2169
	Bi-LSTM with attention	0.48	0.51	0.44	0.4018	0.5638	0.6542
	BERT	0.51	0.45	0.81	0.7512	0.6512	0.5973
	GPT 2	0.42	0.75	0.69	0.6987	0.6945	0.6395
	CANBLWO	0.68	0.72	0.71	0.7629	0.7132	0.6943

$$BLEU_n = \frac{\left(\sum\left(\frac{m_i}{g_i}\right)\right)}{n * \exp(\min(0, 1 - \frac{r}{l})} \quad (69)$$

Equation (69) calculates the BLEU score, where n represents the BLEU type such as n=1 is BLEU-1, n=2 is BLEU-2, n=3 is BLEU-3, and n=4 is BLEU-4. In addition, m_i represents the count of N-grams that match between the reference text and the generated, where g_i is the count of N-grams in the generated text, and the length of reference text is represented as r, and the length of the generated text is denoted as l. Table 4 represents the different BLEU scores. It shows the comparative performance of text generation of other existing deep-learning language models. The proposed model, CANBLWO, emerges as the top performer across most metrics, achieving the highest scores in BLEU-2 (0.71), BLEU-3 (0.79), METEOR (0.7629), ROUGE_L (0.7112), and CIDEr (0.6949), showcasing its exceptional text generation capabilities. BERT and GPT2 also excel, with BERT leading in METEOR (0.7351) and GPT2 in ROUGE_L (0.7393). Bi-LSTM with attention competes well in various metrics, particularly BLEU-2 (0.41) and ROUGE_L (0.5321). In contrast, RNN, LSTM, Bi-LSTM attention BERT and GPT-2 lower scores across all metrics.

Table 4 shows the performance comparison on the Wiki-Bio, CoNaLa, IMDB and Gigawords dataset, where we have seen the CANBLWO model represents a good machine-generated text quality score as compared to other models such as RNN, LSTM, Bi-LSTM, Bi-LSTM with attention model, BERT and GPT-2. Some of the other variations of evaluation metrics are below.

- Recall-Oriented Understudy for Gisting Evaluation (ROUGE): this algorithm is the most promising evaluation metric for the machine generation text [24]. It has been used to compare the reference text and generated text. ROUGE calculates the overlapping between references and generated text. The most commonly used ROUGE scores are ROUGE-L, ROUGE-N, and ROUGE-W.
- ROUGE-L: it calculates the longest matching sequence of words based on the Longest Common Subsequence (LCS) algorithm. To evaluate our generated text, we use the ROUGE-L matrices.

The ROUGE metric calculates the recall of N-grams, where n can be 1 (unigrams), 2 (bigrams), or 3 (trigrams). The mathematical expression for ROUGE-n is:

$$ROUGE_n = \frac{n_g}{n_f} \quad (70)$$

Equation (70) is used to calculate the ROUGE score, where n_g represents the total number of words in generated text and n_f is the number N-gram in the reference text. Next, we have considered another evaluation metric namely METEOR.

METEOR: This metric evaluates the quality of machine-generated text based on how well it matches the reference text [7]. Algorithm (4) represent the calculation of METEOR involves several steps:

Algorithm 4: METEOR Score Calculation.

Input: Machine Generated Text T_{gen}
Output: Evaluation Score METEOR
Step 1. Tokenization Tokens=tokenize(T_{gen})
Step 2. N-gram: N-grams=align_ngrams(Tokens,n)
Step 3. Harmonic mean of precision calculation:

$$P = \frac{1}{1/\sum_{i=1}^{n} p_i}$$

Step 4. Harmonic mean of recall calculation $R = \frac{1}{1/\sum_{i=1}^{n} R_i}$

Step 5. Calculate METEOR score by using step4 and step 5

$$METEOR = \frac{10.P.R}{R + 9.P}$$

Step 6. END

Equation (71) represents the METEOR calculation, where the unaligned word is represented as U, S represents total number of words in reference text, precision alignment is represented by P, and recall alignment is denoted as R.

$$METEOR = \left(1 - \left(\frac{|U|}{|S|}\right)\right) * \frac{2 * P * R}{(P + R)} \quad (71)$$

Table 2 represents, the performance comparison of different evaluation metrics score on various neural network models such as RNN, LSTM, Bi-LSTM, Bi-LSTM with attention, BERT and GPT-2. CoNaLa Dataset is used as a reference text. Table 2, shows the performance of various models across multiple text generation metrics. Our proposed model CANBLWO here also emerges as the top-performing model, achieving high scores in BLEU-2 (0.68), BLEU-3 (0.79), BLEU-4 (0.69), METEOR (0.7629), and ROUGE_L (0.7132). BERT also performs well with scores in BLEU-4 (0.61) and METEOR (0.7512). GPT2 excels in BLEU-3 (0.75), while Bi-LSTM with attention achieves a score in ROUGE_L (0.5638). On the other hand, RNN, LSTM, and Bi-LSTM exhibit comparatively lower scores across most metrics

- Consensus-based Image Description Evaluation (CIDEr): it calculate the similarity between a candidate text and reference text. It checked the presence of specific words, the order in which they appear, and the relationships between them [53].

The Algorithm (5) represent the CIDEr score calculation for our models is as follows:

Algorithm 5: CIDEr Score Calculation.

Input Machine Generated Text T_{gen}
Output Evaluation Score CIDEr
Step 1. Computed TF-IDF weights

$$TF\text{-}IDF_{gen} = compute\_tfidf(T_{gen})$$
$$TF\text{-}IDF_{ref} = compute\_tfidf(T_{ref})$$

Step 2. Cosine similarity

$$cos(m, n) = \frac{m\,.\,n}{||m|| \, * \, ||n||}$$

Step 3. Calculated geometric mean of cosine similarity

$$GeomMean(cos) = \left(\prod_{i=1}^{n} cos(m_i, n_i)\right)^{1/N}$$

Step 4. IDF weight for cos(m, n) for the rare words

$$IDF(w) = log\left(\frac{N_{doc}}{\sum_{d \in docs} I(w \in d)}\right)$$

Step 5. Take exponentially weighted cos(m, n)

$$ExpCos(m, n) = e^{cos(m,n)}$$

Step 6. Got CIDEr score

$$CIDEr = GeomMean(IDF(w).ExpCos(m, n))$$

Step 7. End

Algorithm (5) is a stepwise algorithm to generate the score of CIDEr. Here *cos*(*m*, *n*) is cosine similarity and *m.n* is the dot product of a vector, ||*m*|| * ||*n*|| is the cross product of m, and n vectors where ||*m*|| and ||*n*|| is the length of m and *n*.

5. Conclusions

In this paper, we demonstrated how our proposed model has performed better in terms of contextual text generation tasks. The first part of the proposed model CAN has focussed on every important part of input data to create the feature map. Bi-LSTM has used such feature maps and generated human-like text. Wiki-Bio, CoNaLa, IMDB and Gigawords datasets have been used for the model training and their sentences have been used as the reference text for testing the quality of the machine generated text. Afterwards, WOA is applied to optimize our proposed model's outcomes. We compared other existing models such as RNN, LSTM, Bi-LSTM, Bi-LSTM with the attention model, and few of the popular LLMs such as BERT and GPT-2. BERT and GPT-2 have shown excellent results in larger datasets, but in smaller dataset, our proposed model CANBLWO performed well. In the future, we plan to consider other Indian regional languages, such as Hindi, Bengali, and Tamil, for text generation. Besides, we will explore other evaluation metrics such as BERT score, perplexity, and latent semantics. The proposed model can be incorporated as a software application so that it can be used by users to generate semantically enhanced text.

References

[1] Abadi M., Barham P., Chen J., Chen Z., and Davis A., "TensorFlow: A System for Large-Scale Machine Learning," *in Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, Savannah, pp. 265-283, 2016. https://dl.acm.org/doi/10.5555/3026877.3026899

[2] Abujar S., Masum A., Chowdhury S., Hasan M., and Hossain S., "Bengali Text Generation Using Bi-Directional RNN," *in Proceedings of the 10th International Conference on Computing, Communication and Networking Technologies*, pp. 1-5, Kanpur, 2019. DOI:10.1109/ICCCNT45670.2019.8944784

[3] Alqarni M., "Embedding Search for Quranic Texts Based on Large Language Models," *The International Arab Journal of Information Technology*, vol. 21, no. 2, pp. 243-256, 2024. https://doi.org/10.34028/iajit/21/2/7

[4] Ayana., Chen Y., Yang C., Liu Z., and Sun M., "Reinforced Zero-Shot Cross-Lingual Neural Headline Generation," *Transactions on Audio, Speech, and Language Processing*, vol. 28, no. 12, pp. 2572-2584, 2020. DOI:10.1109/TASLP.2020.3009487

[5] Bai Y., Li Z., Ding N., Shen Y., and Zheng H.,

"Infobox-to-Text Generation with Tree-Like Planning Based Attention Network," *in Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence*, Yokohama, pp. 3773-3779, 2021. https://dl.acm.org/doi/abs/10.5555/3491440.3491962

[6] Balas V., Roy S., Sharma D., and Samui P., *Handbook of Deep Learning Applications*, Springer, 2019. https://doi.org/10.1007/978-3-030-11479-4

[7] Banerjee S. and Lavie A., "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments," *in Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Michigan, pp. 65-72, 2005. https://aclanthology.org/W05-0909

[8] Bao J., Tang D., Duan N., Yan Z., Zhou M., and Zhao T., "Text Generation from Tables," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 27, no. 2, pp. 311-320, 2019. DOI:10.1109/TASLP.2018.2878381

[9] Barros C., Vicente M., and Lloret E., "To What Extent does Content Selection Affect Surface Realization in the Context of Headline Generation?," *Computer Speech and Language*, vol. 67, pp. 101179, 2021. https://doi.org/10.1016/j.csl.2020.101179

[10] Cao J., "Generating Natural Language Descriptions from Tables," *IEEE Access*, vol. 8, pp. 46206-46216, 2020. DOI:10.1109/ACCESS.2020.2979115

[11] Chen X., Jin P., Jing S., and Xie C., "Automatic Detection of Chinese Generated Essayss Based on Pre-Trained BERT," *in Proceedings of the IEEE 10th Joint International Information Technology and Artificial Intelligence Conference*, Chongqing, pp. 2257-2260, 2022. DOI:10.1109/ITAIC54216.2022.9836571

[12] Dethlefs N., Schoene A., and Cuayáhuitl H., "A Divide-and-Conquer Approach to Neural Natural Language Generation from Structured Data," *Neurocomputing*, vol. 433, pp. 300-309, 2021. DOI:10.1016/j.neucom.2020.12.083

[13] Devlin J., Chang M., Lee K., and Toutanova K., "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *in Proceedings of the NAACL-HLT Association for Computational Linguistics*, Minneapolis, pp. 4171-4186, 2019. https://aclanthology.org/N19-1423.pdf

[14] Ding J., Li Y., Ni H., and Yang Z., "Generative Text Summary Based on Enhanced Semantic Attention and Gain-Benefit Gate," *IEEE Access*, vol. 8, pp. 92659-92668, 2020. DOI:10.1109/ACCESS.2020.2994092

[15] Diwan C., Srinivasa S., Suri G., Agarwal S., and Ram P., "AI-based Learning Content Generation and Learning Pathway Augmentation to Increase Learner Engagement," *Computers and Education: Artificial Intelligence*, vol. 4, pp. 100110, 2023. https://doi.org/10.1016/j.caeai.2022.100110

[16] Dixit U., Mishra A., Shukla A., and Tiwari R., "Texture Classification Using Convolutional Neural Network Optimized with Whale Optimization Algorithm," *SN Applied Sciences*, vol. 1, no. 6, pp. 1-11, 2019. DOI:10.1007/s42452-019-0678-y

[17] Faille J., Gatt A., and Gardent C., "The Natural Language Generation Pipeline, Neural Text Generation and Explainability," *in Proceedings of the 2nd Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence*, Dublin, pp. 16-21, 2020. https://hal.science/hal-03046206

[18] Gharehchopogh F. and Gholizadeh H., "A Comprehensive Survey: Whale Optimization Algorithm and its Applications," *Swarm and Evolutionary Computation*, vol. 48, pp. 1-24, 2019. https://doi.org/10.1016/j.swevo.2019.03.004

[19] Jing L., Song X., Lin X., Zhao Z., Zhou W., and Nie L., "Stylized Data-to-Text Generation: A Case Study in the E-Commerce Domain," *ACM Transactions on Information Systems*, vol. 42, no. 1, pp. 1-24, 2023. https://doi.org/10.1145/3603374

[20] Joseph F., Nonsiri S., and Monsakul A., *Advanced Deep Learning for Engineers and Scientists: A Practical Approach*, Springer, 2021. https://link.springer.com/chapter/10.1007/978-3-030-66519-7_4

[21] Kumari S. and Pushphavati T., *Computational Methods and Data Engineering*, Springer, 2023. https://doi.org/10.1007/978-981-19-3015-7_8

[22] Lebret R., Grangier D., and Auli M., "Generating Text from Structured Data with Application to the Biography Domain," *arXiv Preprint*, vol. arXiv:1603.07771, pp. 1-10, 2016. https://www.semanticscholar.org/reader/cc6ef7cebf340606cb9c2f374474f567880fab38

[23] Lee J. and Hsiang J., "Patent Claim Generation by Fine-Tuning OpenAI GPT-2," *World Patent Information*, vol. 62, pp. 101983, 2020. DOI:10.1016/j.wpi.2020.101983

[24] Lin C., "ROUGE: A Package for Automatic Evaluation of Summaries," *in Proceedings of the Workshop on Text Summarization Branches out*, Barcelona, pp. 74-81, 2004. https://aclanthology.org/W04-1013

[25] Liu Y., Wang L., Shi T., and Li J., "Detection of Spam Reviews through a Hierarchical Attention Architecture with N-gram CNN and Bi-LSTM," *Information Systems*, vol. 103, no. C, pp. 101865, 2021. DOI:10.1016/j.is.2021.101865

[26] McKeown K., *Text Generation*, Cambridge University Press, 1992.

https://books.google.jo/books/about/Text_Generation.html?hl=fr&id=Ex6xZlxvUywC&redir_esc=y

[27] Mirjalili S. and Lewis A., "The Whale Optimization Algorithm," *Advances in Engineering Software*, vol. 95, pp. 51-67, 2016. https://doi.org/10.1016/j.advengsoft.2016.01.008

[28] Mukherjee V., Mukherjee A., and Prasad D., *Handbook of Research on Predictive Modeling and Optimization Methods in Science and Engineering, IGI Global*, 2018. DOI:10.4018/978-1-5225-4766-2.ch023

[29] Niculescu M., Ruseti S., and Dascalu M., "RoGPT2: Romanian GPT2 for Text Generation," *in Proceedings of the IEEE 33rd International Conference on Tools with Artificial Intelligence*, Washington, pp. 1154-1161, 2021. DOI:10.1109/ICTAI52525.2021.00183

[30] Palivela H., "Optimization of Paraphrase Generation and Identification Using Language Models in Natural Language Processing," *International Journal of Information Management Data Insights*, vol. 1, no. 2, pp. 100025, 2021. https://doi.org/10.1016/j.jjimei.2021.100025

[31] Papineni K., Roukos S., Ward T., and Zhu W., "BLEU: A Method for Automatic Evaluation of Machine Translation," *in Proceedings of the 40th Annual Meeting on Association for Computational Linguisti*, Philadelphia, pp. 311-318, 2002. DOI:10.3115/1073083.1073135

[32] Pawade D., Sakhapara A., Jain M., Jain N., and Gada K., "Story Scrambler-Automatic Text Generation Using Word Level RNN-LSTM," *International Journal of Information Technology and Computer Science*, vol. 10, no. 6, pp. 44-53, 2018. DOI: 10.5815/ijitcs.2018.06.05

[33] Peters M., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., and Zettlemoyer L., "Deep Contextualized Word Representations," *arXiv Preprint*, vol. arXiv:1802.05365, pp. 1-15, 2018. http://arxiv.org/abs/1802.05365

[34] Qu Y., Liu P., Song W., Liu L., and Cheng M., "A Text Generation and Prediction System: Pre-Training on New Corpora Using BERT and GPT-2," *in Proceedings of the IEEE 10th International Conference on Electronics Information and Emergency Communication*, Beijing, pp. 323-326, 2020. DOI:10.1109/ICEIEC49280.2020.9152352

[35] Rahman M., Watanobe Y., and Nakamura K., "A Bidirectional LSTM Language Model for Code Evaluation and Repair," *Symmetry*, vol. 13, no. 2, pp. 1-15, 2021. https://doi.org/10.3390/sym13020247

[36] Rasheed A., San O., and Kvamsdal T., "Digital Twin: Values, Challenges and Enablers from a Modeling Perspective," *IEEE Access*, vol. 8, pp. 21980-22012, 2020. DOI:10.1109/ACCESS.2020.2970143

[37] Ren Y., Hu W., Wang Z., Zhang X., Wang Y., and Wang X., "A Hybrid Deep Generative Neural Model for Financial Report Generation," *Knowledge-based System*, vol. 227, pp. 107093, 2021. DOI: 10.1016/j.knosys.2021.107093

[38] Roy S., Mallik A., Gulati R., Obaidat M., and Krishna P., "A Deep Learning Based Artificial Neural Network Approach for Intrusion Detection," *in Proceedings of the 3rd International Conference on Mathematics and Computing*, Haldia, pp. 44-53, 2017. https://link.springer.com/chapter/10.1007/978-981-10-4642-1_5

[39] Santhanam S., "Context Based Text-Generation Using LSTM Networks," *arXiv Preprint*, vol. arXiv:2005.00048, pp. 1-10, 2020. http://arxiv.org/abs/2005.00048

[40] Seifossadat E. and Sameti H., "Stochastic Data-to-Text Generation Using Syntactic Dependency Information," *Computer Speech and Language*, vol. 76, pp. 101388, 2022. https://doi.org/10.1016/j.csl.2022.101388

[41] Selva Birunda S. and Kanniga Devi R., *Innovative Data Communication Technologies and Application*, Springer, 2021. https://doi.org/10.1007/978-981-15-9651-3_23

[42] Seo H., Jung S., Jung J., Hwang T., Namgoong H., and Roh Y., "Controllable Text Generation Using Semantic Control Grammar," *IEEE Access*, vol. 11, pp. 26329-26343, 2023. DOI:10.1109/ACCESS.2023.3252017

[43] Sha L., Mou L., Liu T., Poupart P., Li S., Chang B., and Sui Z., "Order-Planning Neural Text Generation from Structured Data," *in Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, pp. 5414-5421, 2018. https://cdn.aaai.org/ojs/11947/11947-13-15475-1-2-20201228.pdf

[44] Shedko A., "Semantic-Map-based Assistant for Creative Text Generation," *Procedia Computer Science*, vol. 123, pp. 446-450, 2018. https://doi.org/10.1016/j.procs.2018.01.068

[45] Shukla A., Das T., and Roy S., "TRX Cryptocurrency Profit and Transaction Success Rate Prediction Using Whale Optimization-based Ensemble Learning Framework," *Mathematics*, vol. 11, no. 11, pp. 1-27, 2023. DOI:10.3390/math11112415

[46] Sutskever I., Vinyals O., and Le Q., "Sequence to Sequence Learning with Neural Networks," *in Proceedings of the 27th International Conference on Neural Information Processing Systems*, Montreal, pp. 3104-3112, 2014. https://dl.acm.org/doi/10.5555/2969033.2969173

[47] Thoppilan R., De Freitas D., Shazeer J., and Kulshreshtha A., "LaMDA: Language Models for Dialog Applications," *arXiv Preprint*, vol.

arXiv:2201.08239, pp. 1-46, 2022. http://arxiv.org/abs/2201.08239

[48] Tiwari S., Khandelwal S., and Roy S., "E-Learning Tool for Japanese Language Learning through English, Hindi and Tamil: A Computer Assisted Language Learning Based Approach," *in Proceedings of the 3rd International Conference on Advanced Computing*, Chennai, pp. 52-55, 2011. DOI:10.1109/ICoAC.2011.6165218

[49] Turki T. and Roy S., "Novel Hate Speech Detection Using Word Cloud Visualization and Ensemble Learning Coupled with Count Vectorizer," *Applied Sciences*, vol. 12, no. 13, pp. 1-13, 2022. https://doi.org/10.3390/app12136611

[50] Van Deemter K., Theune M., and Krahmer E., "Real Versus Template-based Natural Language Generation: A False Opposition?," *Computational Linguistics*, vol. 31, no. 1, pp. 15-24, 2005. https://doi.org/10.1162/0891201053630291

[51] Van der Lee C., Krahmer E., and Wubben S., "Automated Learning of Templates for Data-to-Text Generation: Comparing Rule-based, Statistical and Neural Methods," *in Proceedings of the 11th International Conference on Natural Language Generation*, Tilburg, pp. 35-45, 2018. https://aclanthology.org/W18-6504

[52] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser L., and Polosukhin I., "Attention is all you Need," *in Proceedings of the 31st Conference on Neural Information Processing Systems*, Long Beach, pp. 5999-6010, 2017. https://dl.acm.org/doi/10.5555/3295222.3295349

[53] Vedantam R., Zitnick C., and Parikh D., "CIDEr: Consensus-based Image Description Evaluation," *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, pp. 4566-4575, 2015. DOI:10.1109/CVPR.2015.7299087

[54] Wang H., Hsiao W., and Chang S., "Automatic Paper Writing Based on a RNN and the TextRank Algorithm," *Applied Soft Computing*, vol. 97, pp. 106767, 2020. DOI:10.1016/j.asoc.2020.106767

[55] Wang Z., Cuenca G., Zhou S., Xu F., and Neubig G., "MCoNala: A Benchmark for Code Generation from Multiple Natural Languages," *arXiv Preprint*, vol. arXiv:2203.08388, pp. 1-9, 2023. https://doi.org/10.48550/arXiv.2203.08388

[56] Wei M. and Zhang Y., "Natural Answer Generation with Attention over Instances," *IEEE Access*, vol. 7, pp. 61008-61017, 2019. DOI:10.1109/ACCESS.2019.2904337

[57] Yakhchi S., Behehsti A., Ghafari S., Razzak I., Orgun M., and Elahi M., "A Convolutional Attention Network for Unifying General and Sequential Recommenders," *Information Processing and Management*, vol. 59, no. 1, pp. 102755, 2022. https://doi.org/10.1016/j.ipm.2021.102755

[58] Yang S., Liu Y., Feng D., and Li D., "Text Generation from Data with Dynamic Planning," *IEEE/ACM Trans, Audio, Speech, Lang Process*, vol. 30, pp. 26-34, 2021. DOI:10.1109/TASLP.2021.3129346

[59] Yin P., Deng B., Chen E., Vasilescu B., and Neubig G., "Learning to Mine Aligned Code and Natural Language Pairs from Stack Overflow*," in Proceedings of the 15th International Conference on Mining Software Repositories*, Gothenburg, pp. 476-486, 2018. https://doi.org/10.1145/3196398.319640

[60] Zhang R., Wang Z., Yin K., and Huang Z., "Emotional Text Generation Based on Cross-Domain Sentiment Transfer," *IEEE Access*, vol. 7, pp. 100081-100089, 2019. DOI:10.1109/ACCESS.2019.2931036

[61] Zhao H., Lu J., and Cao J., "A Short Text Conversation Generation Model Combining BERT and Context Attention Mechanism," *International Journal of Computational Science and Engineering*, vol. 23, no. 2, pp. 136-144, 2020. DOI: 10.1504/IJCSE.2020.110536

[62] Zhao J., Zhan Z., Li T., Li R., Hu C., Wang S., and Zhang Y., "Generative Adversarial Network for Table-to-Text Generation," *Neurocomputing*, vol. 452, pp. 28-36, 2021. DOI:10.1016/j.neucom.2021.04.036

Abhishek Kumar Pandey is Research scholar at Vellore Institute of Technology. He is currently pursuing a Ph.D. degree in generative AI and large language model with the University of Vellore Institute, of Technology. He received the B.E. degree in computer science engineering and the M.tech degree in Computer Science engineering from the R.G.P.V University of Bhopal, India, in 2017 and 2019, respectively. His current research interests include Natural Language Processing, Text Generation, Deep Learning, and Sequential Models.

Sanjiban Sekhar Roy is a Professor with the School of Computer Science and Engineering, Vellore Institute of Technology. He uses deep learning and machine learning techniques to solve many complex engineering problems, especially which are related to imagery. He has vast experience in research, especially in the field of advanced machine learning and deep learning. He is specialized in ML and deep convolutional neural networks towards solving various image related complex problem and various engineering problems, such as computational biology, civil, and energy inspired problems. He also has edited special issues for journals and many books with reputed international publishers, such as Elsevier, Springer, and IGI Global. Very recently, the Ministry of National Education, Romania, in collaboration with Faculty of Engineers (‘‘Aurel Vlaicu’’ University of Arad), Romania, has awarded him the ‘‘Diploma of Excellence’’ as a sign of appreciation for the special achievements obtained in the scientific research activity, in 2019.