

# Multi-layered attentional peephole convolutional LSTM for abstractive text summarization

Md. Motiur Rahman<sup>1</sup>  | Fazlul Hasan Siddiqui<sup>2</sup> 

<sup>1</sup>Chattogram Veterinary and Animal Sciences University, Chattogram, Bangladesh

<sup>2</sup>Dhaka University of Engineering and Technology, Dhaka, Bangladesh

## Correspondence

Md. Motiur Rahman, Chattogram, Chattogram Veterinary and Animal Sciences University, Bangladesh.  
Email: motiur@cvasu.ac.bd

## Abstract

Abstractive text summarization is a process of making a summary of a given text by paraphrasing the facts of the text while keeping the meaning intact. The manmade summary generation process is laborious and time-consuming. We present here a summary generation model that is based on multilayered attentional peephole convolutional long short-term memory (MAPCoL; LSTM) in order to extract abstractive summaries of large text in an automated manner. We added the concept of attention in a peephole convolutional LSTM to improve the overall quality of a summary by giving weights to important parts of the source text during training. We evaluated the performance with regard to semantic coherence of our MAPCoL model over a popular dataset named CNN/Daily Mail, and found that MAPCoL outperformed other traditional LSTM-based models. We found improvements in the performance of MAPCoL in different internal settings when compared to state-of-the-art models of abstractive text summarization.

## KEYWORDS

abstractive text summarization, convolutional long short-term memory, deep neural network, long short-term memory, sequence to sequence modeling

## 1 | INTRODUCTION

The process of shortening a large text while keeping the original meaning intact is known as text summarization. A well-organized and concise summary helps humans to obtain the overall scenario of a long text quickly with little labor. There are two forms of text summarization: extractive and abstractive [1]. Extractive summarization generates a summary by taking a subset of sentences or phrases from the original text. It does not generate new sentences, nor does it paraphrase any existing sentences [2,3]. Extractive text summarization has become obsolete with the enhancement of technology. By contrast, abstractive text summarization is

a process of summary generation that paraphrases the given text and preserves the salient meaning of the source text in the summary text [4]. Researchers have recently been drawn to this alternative approach of extractive text summarization. Despite notable success in automated text summarization, conventional methods are unable to resolve all the difficulties associated with text summarization. The quality of a generated summary is measured on the basis of semantic similarity and syntactic structure. The conventional methods, however, focus on one factor only: either semantic similarity or syntactic structure [5]. Extractive text summarization is well-suited for ensuring the syntactic structure during summary generation, but the generated summary is not semantically coherent.

By contrast, abstractive text summarization fails to ensure the syntactic structure of the generated summary, but it is effective in maintaining the semantic coherence [6].

The reason behind the semantic coherence of abstractive text summarization is accomplished by learning the relations between words and choosing the right words to keep the summary semantically relevant. The working procedure of abstractive text summarization is very similar to sequence-to-sequence modeling as both of them take a sequence of text as an input, and a text sequence is generated as a final summary [7]. Here, the modeling of sequence-to-sequence follows the recurrent neural network (RNN) concept, in which a text sequence is provided as an input and looks after each element of that given sequence to determine the next element of output. The long short-term memory (LSTM) is a well-known framework of sequence-to-sequence modeling [8]. It solves many natural-language-based problems such as machine translation and speech recognition. The task of summary generation from a text is not as easy as machine translation because of the variable length of the source text and its summary text [9]. Hence, the mapping between a source text and its summary text is challenging during summary generation. Most of the existing LSTM-based abstractive text summarization models use the traditional architecture of LSTM. One variation of LSTM is known as the peephole convolutional LSTM (PCLSTM), which heavily depends on previous memory-cell content to determine the content of the memory of the current cell [10]. The more the abstractive summary generation procedures are dependent on the previous memory cell content, the more effective the solution that is found from these procedures. This study, introduces a multilayered attentive peephole convolutional LSTM-based (MAPCoL) model for abstractive text summarization.

In this study, we develop an abstractive text summarization model MAPCoL based on the PCLSTM with an attention strategy to generate a summary from a long text. The model is developed by incorporating the multilayer of PCLSTM. The multilayer of PCLSTM is used to capture the complete interaction between the input and output of the model. This multilayer architecture of our model helps to generate a syntactically well-formed summary. The proposed model is implemented by using Keras with a Tensorflow backend, which is a library of Python. The model is applied over a popular dataset to test its versatility. We also compare the performance of the MAPCoL model with state-of-the-art models of abstractive text summarization by tuning internal settings.

This paper is divided into five sections. The first section provides a brief overview of related works. The second section examines the limitations and benefits of traditional LSTM and PCLSTM in summary generation. A new methodology and its evaluation criteria are described in the third section. The next chapter describes the obtained results and the reasons behind those results. Our conclusions and future

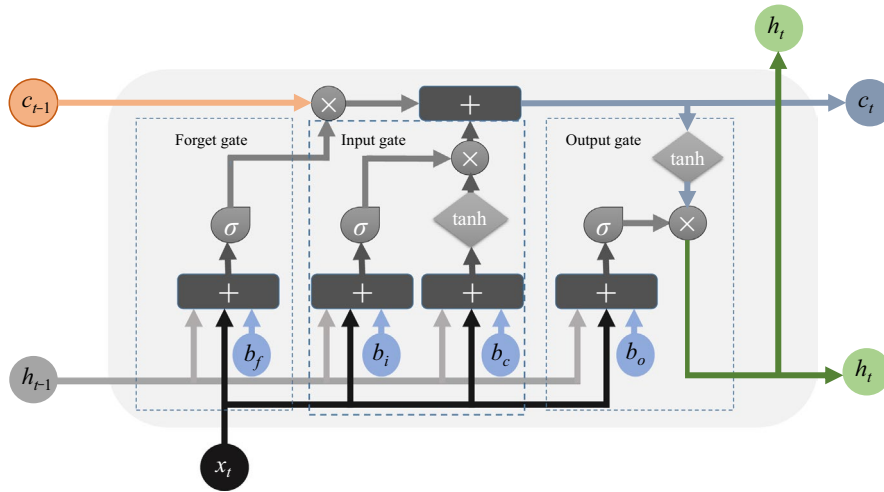
scope of this work are discussed in the last section of this paper.

## 2 | RELATED WORK

Summary generation, especially abstractive summary generation, has become popular with the advancement of machine learning. Although a large amount of research has been conducted to develop an efficient and effective model for extractive text summarization, the number of studies that examined abstractive summarization is very small. Some works that focused on both summarization approaches are illustrated below.

Several neural-network-based methods have been developed for extractive text summarization. A data-driven approach using a feedforward neural network for text summarization is shown in Ref. [11]. Narayan showed a reinforcement learning-based algorithm for sentence ranking that optimizes the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score for extractive summarization [12]. A neural model LeadR was developed to produce a probability distribution over positions of sentences. This helps to locate sentences for summary generation [13]. An end-to-end neural network framework is presented for extractive summarization that selects sentences efficiently based on a calculated score for the summary [14]. Recently, summary generation in a data-driven manner has become very popular, and some works to accomplish this task have been conducted. A neural attention mechanism was introduced in Ref. [15] to build sentences and use them for summarization. An RNN-based sequence model, SummaRuNNer, was shown in Ref. [16]. This model generates a summary by considering content, salience, and novelty. In a recent work, LSTM was used for extractive text summarization. A deletion-based LSTM neural network model was used for sentence compression [17,18]. A comparative study showed that RNN-based summarization techniques return better results than traditional extractive summarization techniques [16].

The RNN is also widely used for abstractive text summarization. In several works [19,20], recurrent neural-network-based models with attention were used for abstractive summary generation. An RNN technique with sequence-to-sequence modeling was shown for abstractive text summarization in Ref. [7]. The researchers utilized a bidirectional neural network and incorporated a Gated Recurrent Unit (GRU) to introduce a new summarization model. Recently, the encoder-decoder framework of RNN has drawn interest in abstractive text summarization. An attentional encoder-decoder model based on LSTM was explained for abstractive text summarization in Ref. [21,22]. Here, the encoder-decoder model could be bidirectional or unidirectional. LSTM with a convolutional neural network (CNN) has become very popular in



**FIGURE 1** Traditional LSTM consists of a memory-block, and three controlling gates such as input, forget, and output gates. These gates control the contents of memory-block, and the output of the current state. Symbols and their corresponding meanings are as follows.  $c_{t-1}$  and  $c_t$  are respectively the contents of the previous and the current memory cells,  $h_{t-1}$  and  $h_t$  are respectively the outputs of the previous and the current states,  $x_t$  is an input vector,  $\times$  is a bitwise multiplication,  $+$  is a bitwise summation,  $\tanh$  is a hyperbolic tangent function,  $\sigma$  is a sigmoid function.  $b_f$ ,  $b_i$ ,  $b_c$ , and  $b_o$  are the bias of the different gates

recent times. ATSDL (Abstractive Text Summarization using Deep Learning) [9] is an abstractive text summarization technique that is modeled by combining LSTM and CNN, and considers both the semantic coherence and syntactic structure of sentences. In this model, input is given as phrases in lieu of simple text, and a challenging task is to identify the phrases from the input text [9].

Convolutional LSTM consists of a memory block and three controlling gates: input, forget, and output gates. These gates control the contents of the memory block and the output of the current state. Symbols and their corresponding meanings are as follows.  $c_{t-1}$  and  $c_t$  are, respectively, the contents of the previous and current memory cells;  $h_{t-1}$  and  $h_t$  are, respectively, the outputs of the previous and current states;  $x_t$  is an input vector;  $\times$  is bitwise multiplication;  $+$  is bitwise summation;  $\tanh$  is a hyperbolic tangent function; and  $\sigma$  is a sigmoid function.  $b_f$ ,  $b_i$ ,  $b_c$ , and  $b_o$  are the biases of the different gates.

Convolutional LSTM is a variation of traditional LSTM that has been applied successfully in different types of prediction. A convolutional LSTM model named ConvLSTM was developed for the precipitation nowcasting problem, and its performance was compared with another model based on a fully connected LSTM (FC-LSTM) [23]. Precise correlation among weather data was considered in this model, which assists predictions with better accuracy than the FC-LSTM model. Another work for time-series classification was proposed based on convolutional LSTM [24]. The researchers used fully convolutional networks along with the recurrent neural network unit LSTM for classifying time-series data, and they achieved the highest classification accuracy. The two aforementioned convolutional LSTM-based models

make predictions with more accuracy than traditional LSTM-based models. The configuration of a convolutional LSTM is better suited than the traditional LSTM for capturing long data patterns effectively. This influences the classification and prediction accuracy.

The above discussion indicates that most works were conducted to generate abstractive summaries using traditional LSTMs. However, none of the studies used parameter optimization to obtaining the highest accuracy. However, traditional LSTM has some limitations. For example, it does not always ensure the semantic coherence of a summary. Because the gate of a traditional LSTM has no connection with the previous memory cell. Traditional LSTM also does not guarantee syntactic coherence. Additionally, the performance of the proposed model could be increased by finding the optimal configuration of parameters. Hence, in our experiment, a model for generating abstractive summaries is developed by utilizing multiple layers of PCLSTM. We added the concept of an attention mechanism that assists in the generation of a concise memory. The process of abstractive summary generation is well fitted with the working mechanism of PCLSTM as it can predict the sequence of text efficiently.

### 3 | ABSTRACTIVE TEXT SUMMARIZATION MODELS

Long short-term memorys in various forms with different architectures have become popular for abstractive text summarization. Traditional LSTM, an initial architecture of LSTM [25], is widely used in text summarization. However, it has some limitations, for example,

dependencies between cells are not strong. This can be resolved by using a PCLSTM, another form of LSTM [23]. The details of the traditional LSTM and PCLSTM are described here.

### 3.1 | Traditional LSTM unit

Long short-term memory, a recurrent neural network unit, is capable of holding and remembering values for a specific period of time. A popular form of LSTM known as traditional LSTM is well-suited for solving sequence-to-sequence related problems where a text sequence is given as input and a text sequence is generated as output. The structural design of the traditional LSTM is displayed in Figure 1. This architecture is composed of a memory cell and three gates: forget, input, and output. Here, the memory cell stores data, and the three other gates control the cell states [26].

According to Figure 1, a line is passed through the top of the architecture, which is known as a memory pipe. The input of the memory pipe is the previous memory content  $c_{t-1}$ . The content of the memory pipe is controlled by the three controlling gates. Here, the first operation found in the memory pipe is bitwise multiplication ( $\times$ ) between the previous memory content and the output received from the forget gate. By using a sigmoid function, the output of the forget gate is converted in a range of 0 to 1. If the generated output from the forget gate is near 0, then most of the previous memory content will nearly be forgotten owing to the multiplication between the previous memory content and the output of the forget gate. By contrast, a significant portion of the previous memory content is passed through the memory pipe when the output of the forget gate is close to 1. Bitwise summation ( $+$ ) is the second operation in the memory pipe and merges the contents of the temporary memory, generated by the input gate, with that of the previous memory, which generates the final memory  $c_t$ .

The forget gate is a single-layer neural network that performs several operations and determines the amount of memory content of the previous cell that will be used at the current timestamp. The forget gate receives the output state  $h_{t-1}$  of previous timestamps, an input vector  $x_t$ , and a bias  $b_f$  as input. The forget gate then generates an accumulated output that is passed through a sigmoid activation function to generate the final output  $f_t$  within the range of 0 to 1. The final output  $f_t$  is multiplied by the previous memory content  $c_{t-1}$ . The value of  $f_t$  determines the significance of the old memory content over the current state. Here, the lower value of  $f_t$  restricts the previous memory content to influence the current state, while the higher value of  $f_t$  allows more previous memory content to contribute over the current state.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f). \quad (1)$$

The input gate controls how much the temporary memory, as described earlier, influences the current memory cell. It takes similar input as the forget gate and produces a temporary memory. The temporary memory is the multiplicative result of a sigmoid function and a hyperbolic tangent function. The sigmoid function produces an output  $i_t$  that will be merged with the previous memory. The  $\tanh$  activation function, whose output is between 0 and 1, determines the amount of temporary memory that will be merged with the memory content of the previous timestamp. A temporary memory having a smaller value of  $\tanh$  function makes a lesser contribution to the current memory cell  $c_t$ . By contrast, a temporary memory having a larger value of the  $\tanh$  function makes a greater contribution to the current memory cell.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \quad (2)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c). \quad (3)$$

The output gate determines the content  $h_t$  of the current state at time  $t$ . The inputs taken by this gate are similar to those of other gates, such as the output of the previous timestamp  $h_{t-1}$  and an input vector  $x_t$  with a different bias value  $b_o$ . This produces  $o_t$  as output. The value of  $o_t$  is within the range 0-1 and determines how much of  $c_t$  should be carried out as the output of the current state. When  $o_t = 1$ , then  $h_t = c_t$ , that is, the entire value of  $c_t$  is passed to the next state as  $h_t$ .

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \quad (4)$$

$$h_t = o_t \tanh(c_t). \quad (5)$$

According to the structural design of the traditional LSTM as shown in Figure 1, the controlling gates are not connected to the memory cell of the previous timestamp. Additionally, the output gate of the traditional LSTM remains mostly closed at the time of training. Hence, as mentioned earlier, owing to these demerits and the closure of the output gate, the memory content of the previous timestamp is inaccessible. This problem has an adverse effect on the performance of text summarization. A variation of the LSTM, called the peephole convolutional LSTM, can overcome this problem. These problems can be resolved by utilizing the merits of the PCLSTM.

### 3.2 | Peephole convolutional LSTM unit

Long short-term memory, a recurrent neural network unit, is widely used in sequence-prediction-related problems. There are different architectures of LSTMs for different purposes. The configuration between gates in the

traditional LSTM restrict the utilization of the memory content of previous timestamps at the time of closure of the output gate [23]. This problem in the traditional LSTM can be solved by using an additional dedicated connection between each gate and the memory content of the previous timestamp. An LSTM with this extra connection is known as PCLSTM, and this connection is denoted as a peephole connection. The peephole connection of a PCLSTM enables all three controlling gates to access the memory content of previous cells even when the output gate is closed. The working principle of the PCLSTM is similar to that of the traditional LSTM except for calculations for the additional connections. The architecture of a PCLSTM is shown in Figure 2 and is implemented as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \quad (6)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \quad (7)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \quad (8)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \quad (9)$$

$$h_t = o_t \tanh(c_t). \quad (10)$$

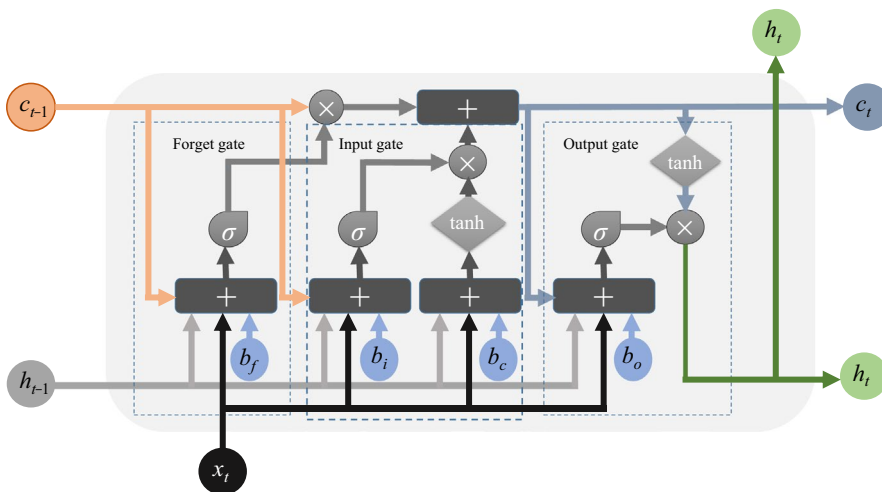
The PCLSTM takes an additional parameter of the memory content of previous cell  $c_{t-1}$  as input along with the inputs of the traditional LSTM, while the traditional LSTM does not take  $c_{t-1}$  as input. The connection with  $c_{t-1}$  in a PCLSTM has a significant impact on the accuracy of sequence-to-sequence prediction tasks. We know that abstractive text summarization is a type of sequence-to-sequence prediction. Hence, this interesting feature of the PCLSTM has motivated the application of this approach to generating abstractive summaries.

### 3.3 | MAPCoL model

In this work, we develop a model for abstractive text summarization using multiple layers of an attentional peephole convolutional LSTM (MAPCoL) that is based on sequence-to-sequence modeling. The architecture of our developed MAPCoL model is shown in Figure 3. The model is composed of two parts: encoder and decoder. Both the encoder and decoder consist of a collection of PCLSTMs, a type of recurrent neural network. The first four PCLSTMs in Figure 3 represent the encoder, and the next four PCLSTMs represent the decoder. The encoder takes a source text as input, tries to understand the input sequences, and produces a small dimensional representation of the source text. This representation is provided to the decoder, which generates another representation, the ultimate output of the encoder-decoder network, after all of the information is decoded.

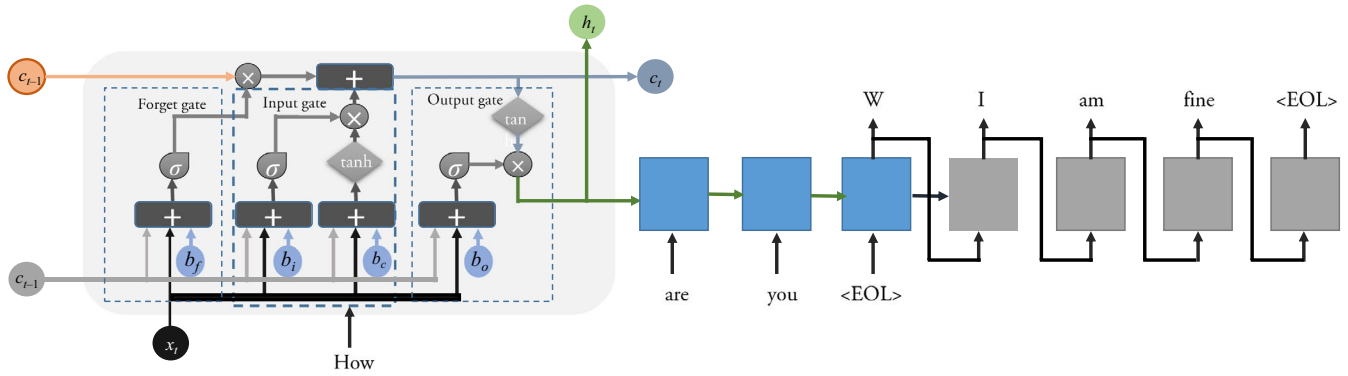
Figure 3 shows that an input ‘‘How are you’’ is passed through the encoder-decoder network. Here, all the words are passed sequentially through the encoder one at a time. After passing the last word through the encoder, the generated representation of the encoder is forwarded to the decoder as input. The decoder generates one word at each time step as output, and the generated word is provided along with other information to generate the next word. After all iterations, the generated output is ‘‘I am fine.’’

The working procedure of MAPCoL is shown in Figure 4. We introduced multiple layers of PCLSTMs in MAPCoL to generate an abstractive summary of a given text. Here, multiple layers of PCLSTMs denote the availability of more than one hidden layer in this model. When the layer size is greater than 3, the overall performance of this model often drops because of overfitting and gradient decay over the layers from the saturated use of activation functions. Hence, two hidden layers are used to build the model. In the beginning, the source text is transformed into

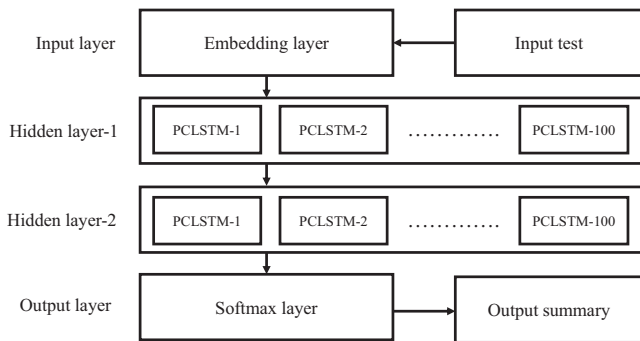


**FIGURE 2** Peephole convolutional LSTM (PCLSTM) is an extension of the traditional LSTM. In a PCLSTM, the previous memory cell  $c_{t-1}$  is linked with each controlling gate, known as peephole connections. These peephole connections allow an extra parameter, memory content  $c_{t-1}$ , to be fed as an input in PCLSTM. This is absent in a traditional LSTM. This extra input in each gate permits access to the memory content of previous cells in any condition





**FIGURE 3** An encoder-decoder network of a sequence-to-sequence model that consists of a PCLSTM block. It takes a sequence of input and produces another sequence as output. The LSTM encoder understands the input sequence and generates a representation that is forwarded to the decoder that eventually generates an output. EOL stands for end of line, which indicates the end of given input



**FIGURE 4** Topology of our MAPCoL model for generating abstractive summaries. Two layers of PCLSTM are used to build the model. The text is embedded using an embedding layer. Each PCLSTM layer has 100 hidden units. The softmax layer removes outliers from the output and maps a vector to a real value

a distributed representation by an embedding layer. The embedding layer performs a simple matrix multiplication that transforms each word into its corresponding word embedded in a distributed representation. A multilayer neural network with a high number of hidden units is used to concatenate these distributed representations. The distributed representation is an intermediate representation of a text that a neural network can process very easily. The given input sequence is transmitted through two hidden layers of PCLSTMs. Each hidden layer of the PCLSTM contains 100 hidden units. When the encoder finishes the encoding of the entire distributed representation, the decoder starts to predict the first word of the summary using an attention mechanism in the softmax layer. The attention mechanism estimates a weight for each word using a probability function that helps remember some important factors of the input. The attention value of a word determines how much attention should be paid to that word during generation of the output as a summary [22]. After passing through the last hidden layer, the average weight of each word is

computed. This is passed through the softmax layer along with the content of the last hidden layer for predicting the next word.

The attentional weight of an input word at position  $t$  is computed when outputting the  $t'$ -th word as follows:

$$a_{y_{t'}}(t) = \frac{\exp(h_{x_t}^T h_{y_{t'}})}{\sum_t \exp(h_{x_t}^T h_{y_{t'}})}, \quad (11)$$

where  $\exp(h_{x_t}^T)$  represents the last hidden layer generated after processing the  $t$ th input word, and  $h_{y_{t'}}$  represents the last hidden layer generated from the current step of decoding. The introduction of attention strategy handles the difficulties of mapping a large source text into a fixed-length output. The softmax layer removes outliers from the output and maps a vector to a real value. The performance of our MAPCoL model was evaluated over the CNN/Daily Mail dataset. A sample of the reference summary (manmade) and system summary is shown in Table 1. Table 1 shows that the quality of the system-generated summary is better in terms of semantic coherence than the reference summary. The syntactic structure of the system-generated summary also is more structured than that of the reference summary.

## 4 | EXPERIMENT

In this study, a model named MAPCoL is developed using a PCLSTM for generating abstractive summaries. Here, we use multilayers of a PCLSTM with 100 hidden units in each layer. We also use an attention mechanism for predicting precise results.

### 4.1 | Experimental setup

The goal of this experiment is to develop a robust and reliable model for abstractive text summarization based on PCLSTM.

TABLE 1 System summary and reference summary

Text	System summary	Reference summary
The Administration of Union Territory Daman and Diu has revoked its order that made it compulsory for women to tie rakhis to their male colleagues on the occasion of Rakshabandhan on August 7. The administration was forced to withdraw the decision within 24 h of issuing the circular after it received flak from employees and was slammed on social media.	Daman and Diu revokes mandatory Rakshabandhan in offices order	Daman and Diu revokes mandatory Rakshabandhan in offices order
Malaika Arora slammed an Instagram user who trolled her for "divorcing a rich man" and "having fun with the alimony." "Her life now is all about wearing short clothes, going to [the] gym or salon, [and] enjoying vacation[s]," the user commented. Malaika responded, "You certainly got to get your damn facts right before spewing shot on me... when you know nothing about me."	Malaika hits people who trolled her for divorcing [a] rich man	Malaika slams [a] user who trolled her for 'divorcing [a] rich man'

Note: The 'text' column represents the input text in our model. The 'system summary' column shows the generated summary from our MAPCoL model. The human generated summary is shown in the 'reference summary' column.

MAPCoL is implemented using TensorFlow backed with Keras, which is a Python library. In our experiment, the batch size and number of epochs have significant roles with regard to performance. The batch size denotes the portion of the dataset that is passed together through the model at a single time. The same data are passed through the model multiple times; this is known as the epoch number. For this experiment, after performing parameter tuning, we select 64 as the batch size and 200 as the epoch number to train and test the model. By using this configuration, we hypothesize that our MAPCoL model produces better accuracy with regard to the syntactic structure and semantic coherence with this configurations.

## 4.2 | Experimental dataset

We evaluated the performance of our MAPCoL model over the CNN/Daily Mail dataset, which utilizes the news from CNN and the Daily Mail newspaper. The dataset contains 286 817 training pairs, 13 368 validation pairs, and 11 487 test pairs of data.

## 4.3 | Evaluation method

Traditionally, the quality of a summarized text is evaluated manually by humans by judging parameters such as conciseness, relevancy, coherence, grammar, and readability [27]. Manual evaluation to determine the quality is time-consuming, difficult, and repetitive. Therefore, researchers developed automated methods to evaluate summaries. Most of these methods are based on the similarity measure between a summary and its original text, but they do not relate the judgment with human judgment [28]. Hence, a recall-oriented method named ROUGE was developed to evaluate the quality of the summary [28]. Here, the system summary and the reference summary (human-generated)

are compared to evaluate the summary quality. The ROUGE score is measured as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in \text{Ref}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \text{Ref}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)}, \quad (12)$$

where  $n$  is the number of consecutive words, Ref is the set of reference summaries,  $\text{Count}(\text{gram}_n)$  counts the number of  $n$  consecutive words of the reference summary, and  $\text{Count}_{\text{match}}(\text{gram}_n)$  estimates  $n$  number of consecutive words matched between the reference summary and system summary. Here, the human-generated summary is denoted as the reference summary, and our model-generated summary is denoted as the system summary. The division of the number of matches between the two referred summaries by the length of the reference summary is called the ROUGE score. We calculate ROUGE-1 and ROUGE-2 scores that define the semantic similarity and syntactic structure, respectively. ROUGE-1 evaluates the quality of a summary by taking single words into account, while ROUGE-2 evaluates the summary quality by considering two consecutive words at a time.

## 5 | RESULT AND DISCUSSION

Deep-learning-based abstractive text summarization has become popular in recent times. We introduced a multi-layered attentional peephole-convolutional-LSTM-based abstractive text summarization model, MAPCoL, for summary generation. The performance of our model was evaluated using recall-oriented techniques called ROUGE-1 and ROUGE-2.

We applied our model to a popular CNN/Daily Mail dataset, and the reported results were compared with the other models, as shown in Table 2. All mentioned models shown in Table 2 were evaluated over the same dataset. In addition, the

**TABLE 2** Performance comparison for MAPCoL model and other models on CNN/Daily Mail dataset

Model	ROUGE-1 (%)	ROUGE-2 (%)
SummaRuNNer	39.60	16.20
Graph-Based	38.10	13.90
feats-lvt2k-2sent-ptr	36.40	17.77
ATSDL	34.90	17.80
MAPCoL	39.61	20.87

Note: ROUGE-1 denotes the quality of system summary by taking single words into consideration, while ROUGE-2 takes two consecutive words into consideration. Here, highlighted values indicate the best results among all models.

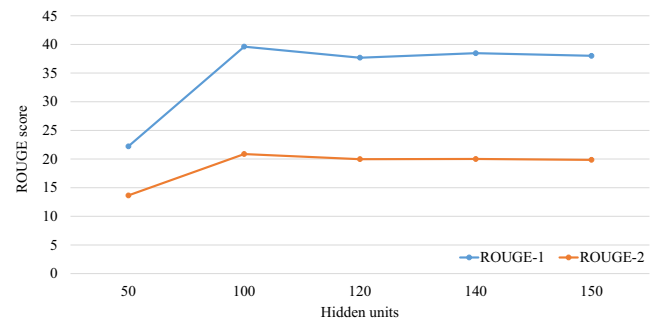
experimental configurations of these models are similar. We calculated the ROUGE score to evaluate the performance of our model. To perform a comparative analysis, the reported results of Table 2 were taken from [9].

A higher ROUGE score indicates better quality of the summary, while the lower ROUGE score indicates the opposite. All models shown in Table 2 are based on two different technologies: our developed MAPCoL is based on the PCLSTM, and the other four models [SummaRuNNer [16], Graph-Based [29], feats-lvt2k-2sent-ptr [7], and ATSDL [9]] are based on the traditional LSTM. Table 2 compares the performance of the PCLSTM-based model with that of the traditional LSTM-based models. Among the traditional LSTM-based models, the SummaRuNNer obtained the highest ROUGE-1 score of 39.6%, and model ATSDL achieved the highest ROUGE-2 score of 17.8%. However, among the five models, the highest ROUGE-1 and ROUGE-2 scores were obtained by the PCLSTM-based model MAPCoL: 39.61% and 20.87%, respectively. The comparison shows that although the improvement of the ROUGE-1 score of the PCLSTM-based model is not large, we found a significant improvement of the ROUGE-2 score of the PCLSTM-based model when compared with the traditional LSTM-based model. We reported on the possible reasons of such accuracy. The utilization of the PCLSTM instead of the traditional LSTM enables us to capture the long-term data dependency very effectively with the help of the memory content of previous timestamps. It is known that the structural quality (ROUGE-2) of a summary depends on better sequence generation in the output, which is significantly influenced by an effectively captured long data pattern. In addition, the incorporation of the multilayered topology of PCLSTM also contributed to achieving better accuracy in this regard.

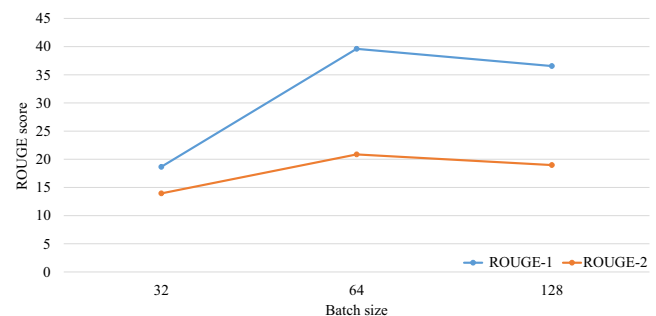
Another goal of this study is to find the configurations where our model works better and those where it does not. Some comparative analyses were conducted by changing the values of parameters to accomplish these tasks. The scores of ROUGE-1 and ROUGE-2 of our MAPCoL model were

observed for different hidden units such as 50, 100, and 150. The behavior of our model in these experiments is shown in Figure 5. We reported on the lowest ROUGE-1 and ROUGE-2 scores when the number of hidden units was 50, and found a sharp increase in the performance when the hidden unit size was increased to 100. However, we found no significant improvement in the performance of MAPCoL when the number of hidden units was higher than 100, and the training time also increased significantly. These experiments suggested that the number of hidden units be kept at 100 in order to achieve the maximum performance of MAPCoL.

The performance of our MAPCoL model was also observed by changing the batch size to 32, 64, and 128. Here, the batch size defines how much volume of the dataset will be passed through the model at a single time. The characteristics of the MAPCoL model with different batch sizes is shown in Figure 6. When the model was trained by keeping the batch size at 32, we reported the lowest ROUGE-1 and ROUGE-2 scores. With an increase of the batch size to 64, the performance of MAPCoL was ameliorated significantly and achieved the highest ROUGE scores. By contrast, when

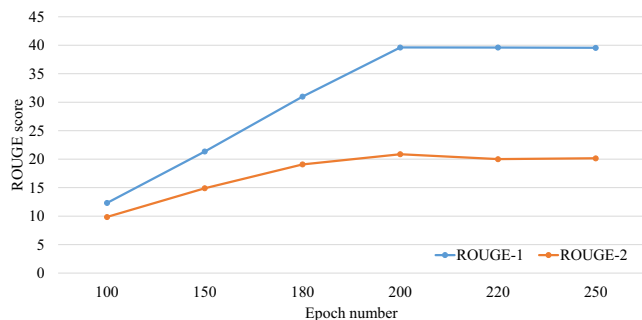


**FIGURE 5** Reported ROUGE scores of our developed model under variant sizes of hidden units. The performance of MAPCoL increases sharply until the number of hidden units is 100. If unstable, the performance keeps improving at a very slow rate after that mark. Note that a greater number of hidden units requires more time to complete the training

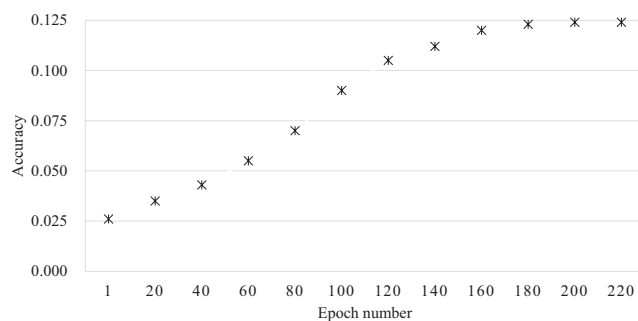


**FIGURE 6** Reported ROUGE scores of our developed model under different batch sizes. Here, the ROUGE score grows rapidly with a change of batch size from 32 to 64, and flattens out afterward. An oversized batch is also increased the training time





**FIGURE 7** Reported ROUGE scores of our developed model under different epoch numbers. The figure shows that the performance of the MAPCoL model increases with an increase in the epoch number from 100 to 200. An epoch number over 200 does not have a significant effect on the performance



**FIGURE 8** Comparison of accuracy gain with epoch numbers. The accuracy of the training data increases visibly with an increase in the epoch number and becomes stable after 200 epochs

the batch size was increased to 128, we reported a notable decline in the performance of MAPCoL. The model behaved in this way because underfitting and overfitting problems occur when the batch sizes were 32 and 128, respectively. This led to poor generalization of the given new data. Additionally, a large batch size adversely increased the training time rather than the performance.

The performance of a machine learning model also depends on the epoch number. Hence, it is essential to determine the optimal epoch number where our model performs best. We observed the performance of MAPCoL from 100 to 250 epochs as shown in Figure 7. We found a linear increase in the ROUGE-1 and ROUGE-2 scores until the epoch number was 200, and no visible improvement afterward. These experiments suggest that the number of hidden units should be 100, the batch size 64, and that the model be trained for 200 epochs.

Different models behave differently over the same dataset. We reported on the accuracy gain of our MAPCoL model to quantify the training accuracy over the CNN/Daily Mail dataset. The accuracy was calculated by comparing the predicted result with the true result. Figure 8 shows that the model was

efficiently fitted with the dataset and that the training accuracy increased with an increasing epoch number until the number was 200. In addition, the accuracy flattened after 200 epochs were completed.

The discussed experiments verified our hypothesis that the PCLSTM-based MAPCoL model outperformed the state-of-the-art models for abstractive text summarization. We found that the peephole connection of the PCLSTM had a significant impact on generating semantically and syntactically coherent abstractive summaries. The conscious choice of using multiple layers of a PCLSTM helped us to effectively capture long data dependency, which contributed significantly to generating concise summaries.

## 6 | CONCLUSION


We introduced a model named MAPCoL for abstractive summary generation using multiple layers of a PCLSTM. The peephole connection and incorporation of multiple layers of PCLSTM enabled our model to generate semantically and syntactically coherent abstractive summaries with higher accuracy than any state-of-the-art models. Most of the developed abstractive summarization models have some limitations that were resolved by our MAPCoL model. It is known that the most of the existing abstractive text summarization techniques generate fixed-length summaries, while our introduced model generates variable-length summaries by taking variable lengths of text as input. We prioritized the semantic and syntactic coherence of the summaries during development of the model. We handled rare words and unwanted symbols while preprocessing a given text. We trained the multilayered peephole convolutional LSTM-based model using this preprocessed data. An important aspect of our model is that it can receive any length of input data and can generate any length of output as a summary. A popular dataset, CNN/Daily Mail, was used to evaluate the performance of MAPCoL, and we compared the performance to other abstractive summarization models that were also evaluated over the same dataset. To determine for which configurations our model performs more effectively, experiments were conducted by changing the values of different parameters, and the optimal configurations were reported.

## CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

## ORCID

Md. Motiur Rahman  <https://orcid.org/0000-0001-7472-4085>

Fazlul Hasan Siddiqui  <https://orcid.org/0000-0003-4825-6315>

## REFERENCES

1. J. Chen and H. Zhuge, *Abstractive text-image summarization using multi-modal attentional hierarchical RNN*, in Proc. Conf. Empirical Methods Natural Language Process (Brussels, Belgium), 2018, pp. 4046–4056.
2. D. Yogatama, F. Liu, and N. A. Smith, *Extractive summarization by maximizing semantic volume*, in Proc. Conf. Empirical Methods Natural Language Process (Lisbon, Portugal), 2015, pp. 1961–1966.
3. P. Mehta, *From extractive to abstractive Summarization: A journey*, in Proc. ACL 2016 Student Research Workshop (Berlin, Germany), 2016, pp. 100–106.
4. P. Li et al., *Deep recurrent generative decoder for abstractive text summarization*, in Proc. Conf. Empirical Methods Natural Language Process (Copenhagen, Denmark), 2017, pp. 2091–2100.
5. P.-E. Genest and G. Lapalme, *Framework for abstractive summarization using text-to-text generation*, in Proc. Workshop Monolingual Text-To-Text Generation (Portland, OR, USA), 2011, pp. 64–73.
6. J. Cheng and M. Lapata, *Neural summarization by extracting sentences and words*, in Proc. Annu. Meeting Association Comput. Linguistics (Berlin, Germany), Mar. 2016, pp. 484–494.
7. R. Nallapati et al., *Abstractive text summarization using sequence-to-sequence rnns and beyond*, in Proc. SIGNLL Conf. Comput. Natural Language Learn. (Berlin, Germany), 2016, pp. 280–290.
8. Y. Zhang, Q. Liu, and L. Song, *Sentence-state LSTM for text representation*, in Proc. Annu. Meeting Association Comput. Linguistics (Melbourne, Australia), 2018, pp. 317–327.
9. S. Song, H. Huang, and T. Ruan, *Abstractive text summarization using LSTM-CNN based deep learning*, *Multimedia Tools Appl.* **78** (2018), 1–19.
10. F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, *Learning precise timing with LSTM recurrent networks*, *J. Machine Learn. Res.* **3** (2003), no. 1, 115–143.
11. A. Sinha, A. Yadav, and A. Gahlot, *Extractive text summarization using neural networks*, arXiv Preprint, CoRR, (2018), abs/1802.10137.
12. S. Narayan, S. B. Cohen, and M. Lapata, *Ranking sentences for extractive summarization with reinforcement learning*, in Proc. Conf. North American Chapter Association Comput. Linguistics: Human Language Technol. (New Orleans, LA, USA), 2018, pp. 1747–1759.
13. T.A. Bohn and C.X. Ling, *Neural sentence location prediction for summarization*, arXiv Preprint, CoRR, 2018, abs/1804.08053.
14. Q. Zhou et al., *Neural document summarization by jointly learning to score and select sentences*, in Proc. Annu. Meeting Association Comput. Linguistics (Melbourne, Australia), 2018, pp. 654–663.
15. S. Tarnpradab, F. Liu, and K. A. Hua, *Toward extractive summarization of online forum discussions via hierarchical attention networks*, arXiv Preprint, CoRR, 2018, abs/1805.10390.
16. R. Nallapati, F. Zhai, and B. Zhou, *Summarunner a recurrent neural network based sequence model for extractive summarization of documents*, arXiv Preprint, CoRR, 2016, abs/1611.04230.
17. L. Wang et al., *Can syntax help? improving an lstm-based sentence compression model for new domains*, in Proc. Annu. Meeting Association Comput. Linguistics (Vancouver, Canada), 2017, pp. 1385–1393.
18. K. Filippova et al., *Sentence compression by deletion with lstms*, in Proc. Conf. Empirical Methods Natural Language Process. (Lisbon, Portugal), 2015, pp. 360–368.
19. A. M. Rush, S. Chopra, and J. Weston, *A neural attention model for abstractive sentence summarization*, arXiv Preprint, CoRR, 2015, abs/1509.00685.
20. S. Chopra, M. Auli, and A. M. Rush, *Abstractive sentence summarization with attentive recurrent neural networks*, in Proc. Conf. North American Chapter Association Comput. Linguistics: Human Language Technol. (San Diego, CA, USA), 2016, pp. 93–98.
21. K. Al-Sabahi, Z. Zuping, and Y. Kang, *Bidirectional attentional encoder-decoder model and bidirectional beam search for abstractive summarization*, arXiv Preprint, CoRR, 2018, abs/1809.06662.
22. K. Lopyrev, *Generating news headlines with recurrent neural networks*, arXiv Preprint, CoRR, 2015, abs/1512.01712.
23. X. Shi et al., *Convolutional LSTM network: A machine learning approach for precipitation nowcasting*, in Proc. Int. Conf. Neural Inf. Process. Syst. (Montreal, Canada), 2015, pp. 802–810.
24. F. Karim et al., *Lstm fully convolutional networks for time series classification*, *IEEE Access* **6** (2018), 1662–1669.
25. C. A. Colmenares et al., *HEADS: Headline generation as sequence prediction using an abstract feature-rich space*, in Proc. Conf. North Am. Chapter Association Comput. Linguistics: Human Language Technol. (Denver, CO, USA), 2015, pp. 133–142.
26. Z. C. Lipton, *A critical review of recurrent neural networks for sequence learning*, arXiv Preprint, CoRR, 2015, abs/1506.00019.
27. J.-P. Ng and V. Abrecht, *Better summarization evaluation with word embeddings for ROUGE*, in Proc. Conf. Empirical Methods Natural Language Process. (Lisbon, Portugal), 2015, pp. 1925–1930.
28. S. Martschat, and K. Markert, *Improving ROUGE for timeline summarization*, in Proc. Conf. Eur. Chapter Association Comput. Linguistics (Valencia, Spain), 2017, pp. 285–290.
29. J. Tan, X. Wan, and J. Xiao, *Abstractive document summarization with a graph-based attentional neural model*, in Proc. Annu. Meeting Association Comput. Linguistics (Vancouver, Canada), 2017, pp. 1171–1181.

## AUTHOR BIOGRAPHIES



**Md. Motiur Rahman** is an assistant professor at the Chattogram Veterinary and Animal Sciences University, Chattogram, Bangladesh. He received his MSC (Engineering) degree in computer science and engineering from Dhaka University of Engineering & Technology, Dhaka, Bangladesh, and his research area was deep learning, and his BSC (Engineering) degree in computer science and engineering from the Patuakhali Science and Technology University, Patuakhali, Bangladesh. As a principal investigator, he is currently working on three small-scale projects based on artificial intelligence funded by the Bangladesh government. His research interests include machine learning, computer vision, data analytics, algorithm optimization, and wireless sensors.



**Fazlul Hasan Siddiqui** is a professor in the Department of Computer Science & Engineering at the Dhaka University of Engineering & Technology, Gazipur, Bangladesh. He received his doctorate in automated planning in the field of artificial intel-

ligence from the Australian National University, Canberra, Australia, in 2016. His doctorate dissertation received the Automated Planning and Scheduling (ICAPS) Best Dissertation Award at the prestigious International Conference in the year 2017. He received his MSC in computer science & engineering from the Bangladesh University of Engineering & Technology, Dhaka, Bangladesh in 2007, and his BSC in computer science & IT from the Islamic University of Technology, Gazipur, Bangladesh in 2002. His key publications include Continuing Plan Quality Optimisation (JAIR, 2015) and Abstractive Text Summarization using Response Surface Method (Symmetry, 2019). His current research interests include automated planning and deep learning in the field of artificial intelligence.