International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Sequence-to-Sequence Abstractive Text Summarization Model for Headline Generation with Attention

Shreya Regundwar[1]*, Radhika Bhagwat[2]*, Sakshi Bhosale[3], Rajlaxmi Chougale[4], Sakshi Abbu[5]

Submitted: 28/01/2024 **Revised**: 06/03/2024 **Accepted**: 14/03/2024

Abstract: Text summarization focuses on creating a brief and concise summary from source text while preserving the main idea and eliminating unnecessary details. Generating summaries through manual efforts by humans is a tedious, tiresome, and expensive process. Hence, this study's objective is to build an automated abstractive text summarizer that can minimize manual efforts and generate concise summaries swiftly. The aim is to develop a text summarizer model using deep learning to form a single-line abstractive summary resembling a headline. It also explores the impact of adjusting the model's hyperparameters on the generated summary to achieve better results. A subset of instances from the Gigaword dataset is utilized to develop the model. The proposed summarizer is a sequence-to-sequence model with an LSTM-driven encoder-decoder architecture. It incorporates a Bahdanau attention mechanism and utilizes the Adam optimizer. Based on experimental analysis and the results obtained after adjusting hyperparameters and selecting the optimal values as final, the proposed architecture attained scores as 24.27, 8.57, and 23.13, for ROUGE-1, ROUGE-2, and ROUGE-L respectively.

Keywords: *Abstractive text summarization, Attention mechanism, Deep learning, Encoder-Decoder architecture, LSTM, Sequence-to-Sequence model, Single sentence summarization, Text preprocessing techniques.*

1. Introduction

In the present scenario, there is abundant textual data, be it online documents, articles, news, or reviews [1]. These are often characterized by lengthy text strings, highlighting the need for effective summarization. Summarization involves generating a concise version of textual information, typically encapsulating the essential details from the source document. Text summarization is categorized into extractive summarization and abstractive summarization. In extractive summarization, summaries are formed solely by selecting and assembling content directly from the source text. This approach is considered simpler as it guarantees grammatical accuracy by copying information directly from the source text [2]. However, extractive summaries may contain unnecessary details. In contrast, abstractive summaries are paraphrased versions [3], they maintain the core notion or context behind the original text. They are generally more readable, Understandable, and coherent compared to extractive summaries.

The usage of deep learning concepts in abstractive text summarization began in 2015 [4], by introducing a model whose basis was the encoder-decoder architecture. To produce a single sentence abstractive summary, this work employed a sequence-to-sequence model (Seq2Seq) incorporating a stack of three bidirectional Long Short-Term Memory layers (Bi-LSTM) in an encoder for processing input text. Additionally, it utilized a single-layered unidirectional Long Short-Term Memory (LSTM) within the decoder with the Bahdanau attention mechanism [5] on the target text. Gigaword dataset [6] formed the foundation for building the model. Features within the input data (news) are grasped by the encoder. Then it produces a context vector that has a definite or constant length. This vector is further transmitted to the decoder and utilized to form a summary in accordance with the article.

The evaluation technique is crucial in assessing the performance of a summarization model. Intrinsic evaluation involves assessment techniques concentrating on metrics like F-score, precision, and recall [7]. BLEU and ROUGE are intrinsic metrics broadly used to assess machine generated-summary [8]. Mostly, ROUGE is employed to evaluate machine-generated text summaries [9], [10], [11], [12]. Thus, in this study, assessment of the model's effectiveness is conducted with the help of the ROUGE score, relying on both the target summaries produced by the model and reference summaries formed by humans.

2. Literature Review

Extractive [13] and abstractive summarization [14] are methods for summarizing text. In extractive text summarization, essential sentences and paragraphs or other elements from the source document are chosen and combined to create a more concise version [14]. Abstractive Text Summarization is typically a challenging

[1,2,3,4,5] *Department of Information Technology, Cummins College of Engineering for Women, Pune, Maharashtra, India*
[1]*ORCID ID: 0009-0005-5674-1320*
[2]*ORCID ID: 0000-0001-5127-7980*
[3]*ORCID ID: 0009-0001-4532-1321*
[4]*ORCID ID: 0009-0000-6639-6237*
[5]*ORCID ID: 0009-0004-5204-9196*
** Corresponding Author Email:*
shreya.regundwar@cumminscollege.in
radhika.bhagwat@cumminscollege.in

and time-intensive process, frequently yielding suboptimal results due to the limitations of computers in comprehending human language accurately [15]. Researches conducted on abstractive text summarization are based on various factors such as types of neural networks employed in models (RNN, LSTM, GRU), the type of input documents i.e. whether the document is multiple or single, the type of output (single line or multiple lines), objectives (query-based, domain-specific, or generic), and performance metrics [1], [15].

We find that abstractive text summarization have been significantly impacted due to introduction of neural Seq2Seq model. Unlike traditional methods, these models provide a more flexible and expressive way of summarizing text [12], [16]. Tian Shil et al. [17] have provided a detailed explanation of the Seq2Seq model. An encoder-decoder model with recurrent neural networks (RNNs) is built by Nallapati et al. [12]. LSTM is used to effectively address the gradient-related issues that are encountered by RNN [18]. Sangita et al. [19] and Panagiotis Kouris et al. [20] have utilized Bi-LSTM within the encoder, further within the decoder unidirectional LSTM is utilized. Wazery et al. [21] observed optimal performance when three layers of Bi-LSTM are utilized within the encoder. The encoder-decoder architecture having RNN with attention, is introduced in the work by Bahdanau et al. [5] and has solidified itself as a standard in abstractive summarization.

For abstractive text summarization, the Gigaword dataset [12], [21], [22], Amazon Fine Food Reviews [23], [11], DUC2003, DUC2004 [21], and the CNN/Daily Mail dataset [2], [24] are among the most frequently utilized datasets. The Gigaword dataset and Amazon fine food reviews dataset are utilized when working on single-sentence summaries. CNN/Daily Mail dataset is taken into account when working on multiple line summaries.

Table 1. Datasets for abstractive summarization

Dataset name	**Line count**	**Text length**	**Summary length**	**Train**	**Test**
Gigaword	single line	31	15	3.8M	1.9K
CNN/ daily mail	multi line	760	55	287K	11K
Amazon fine food reviews	single line	30	8	1M	10K

Table 1 presents a summary of the prevalent datasets used for abstractive text summarization. The table includes information such as the dataset name, line count or number of sentences in summaries, text length (max count of words in text), summary length (max count of words in summary), and count of data entries in both the training and testing datasets.

The assessment of the text summarization system is done with the help of set of metrics known as ROUGE, by comparing model-generated summaries with manually generated summaries [2], [12], [19], [21], [24].

This research study introduces a model designed to perform abstractive summarization of text, focusing on headline generation or single-sentence summaries within the news domain. This approach involved the utilization of Bi-LSTM within the encoder coupled with unidirectional LSTM within the decoder. The Gigaword dataset [6], obtained from Hugging Face, formed the foundation for building, training, and testing the model. Various hyperparameter adjustments were made to explore the impact of these hyperparameters on the generated summaries, to achieve improved results.

3. Method

This section describes the methodology employed in creating an abstractive text summarizer designed for producing English headlines from English news content.

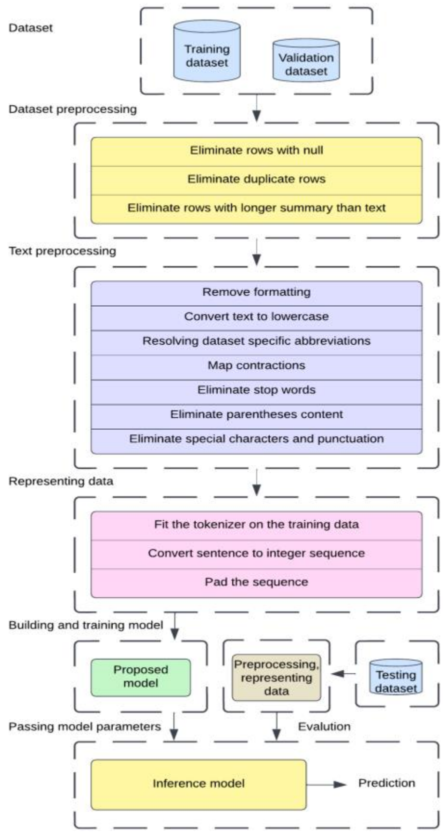

Dataset
Training dataset
Validation dataset
Dataset preprocessing
Eliminate rows with null
Eliminate duplicate rows
Eliminate rows with longer summary than text
Text preprocessing
Remove formatting
Convert text to lowercase
Resolving dataset specific abbreviations
Map contractions
Eliminate stop words
Eliminate parentheses content
Eliminate special characters and punctuation
Representing data
Fit the tokenizer on the training data
Convert sentence to integer sequence
Pad the sequence
Building and training model
Proposed model
Preprocessing, representing data
Testing dataset
Passing model parameters
Evalution
Inference model
Prediction

Fig. 1. Architecture of abstractive text summarization system

The architectural representation in Figure 1 outlines the workflow used to develop our abstractive text summarization system. The initial stages involve dataset and text preprocessing, preparing the input data for an abstractive text summarization system. This system is constructed by employing Gigaword dataset for both training and testing, as well as validation purposes. Trained model’s parameters are passed to the inference model. The testing dataset is utilized to evaluate summaries produced by the model.

3.1. Dataset

In this study, experiments are conducted using the Gigaword dataset [6]. The dataset contains the following splits: training, validation, and testing. Each part has short text and headlines. The training set comprises approximately 3,803,957 instances, the validation dataset consists of 189,651 data points, and the test dataset has 1951 instances. To build and conduct experiments with the model, this study utilized 2,00,000 instances to train the model, 20,000 for validating, and 1,000 for testing purposes.

3.2. Dataset preprocessing

During the data preprocessing phase, duplicate rows were eliminated. Additionally, instances were removed if the news text length exceeded that of their corresponding headlines, or if either the news text or headline was missing.

3.3. Text preprocessing

Text preprocessing [11], [23] involves tasks such as cleaning data, eliminating ambiguities, and organizing textual data to prepare it as input for models. Text data is inherently complex, unstructured and comprises sequences of words, punctuation, symbols, numbers, contractions, markup, and formatting elements. Furthermore, certain components such as punctuation and stop words ('the’, 'such', 'any', 'very', 'am', 'is', etc.) which hold minimal relevance, especially Within the field of abstractive text summarization. The following section details various operations employed in the preprocessing of text data:

3.3.1. Remove formatting

The first step is to eliminate any HTML tags, markup, and formatting elements present in the text.

3.3.2. Convert to lowercase

Text is converted to lowercase [19] to ensure uniformity in processing.

3.3.3. Resolving dataset-specific abbreviations

The Gigaword dataset [6] includes abbreviations such as "fm" for "foreign minister", "dlrs" for "dollars", "pct" for "percent", "mln" for "million", and "gov" for "government". These abbreviations are replaced with their respective full forms during preprocessing.

3.3.4. Map contractions

In the English language, contractions represent shortened forms of word combinations (e.g., "can't" for "can not," "won't" for "will not"). Contraction mapping is performed to ensure semantic clarity and maintain consistency for accurate analysis.

3.3.5. Eliminate stop words

Stop words with little semantic value are removed, to focus on content-carrying words [2].

3.3.6. Eliminate parentheses content

Parentheses content from the source text consists primarily of non-essential details, so it is removed.

3.3.7. Eliminate punctuations, special characters, double quotes, 's

Punctuation, special characters, quotes, and possessive forms ('s) lack substantial semantic meaning and may introduce unwanted noise to the data; hence, they are eliminated.

Additionally, in this paper's dataset, infrequent words had already been replaced with "UNK," and digits were substituted with "#". Instances of the word "unk" were removed from the data as they were considered nonessential. Furthermore, punctuation and special characters, such as "#," were eliminated using regular expressions.

3.4. Representing data

Deep learning models do not inherently comprehend words directly. Instead of using pre-trained embeddings for word representation, the approach here is to train embeddings from scratch when doing the training process of the model [2]. To accomplish this, a Keras tokenizer is employed to convert words into integers. These integers, serving as mappings for words, are then fed into the model. Subsequently, an embedding layer transforms these integers into vectors. To train the abstraction text summarization model efficiently, we set a maximum length for both the news text and summary and performed padding. If the input sequence falls short of the maximum length, we supplement it by adding 0 integers until it attains the prescribed length. Conversely, if the input sequence exceeds the maximum length, it is truncated. This approach ensures uniform processing and handling of variable-length input sequences [3]. Notably, the embedding layer’s parameters are configured as trainable, allowing the vectors representing words to be updated and refined throughout the training process. This paper adopts a word embedding dimensionality of 300 [19].

3.5. Model Architecture

3.5.1. Seq2Seq model

Seq2Seq model [12], [23] transforms the input sequence within the neural network of sequences consisting of letters, words, and phrases. This work aims to develop a deep learning system to produce abstractive summaries using encoder and decoder networks. The encoder incorporates a three-layered structure of

Bi-LSTM, however the decoder consist one layer of unidirectional LSTM, to capture related information for quality improvement of generated summary. Output of the encoder and decoder are inputted into an attention layer (in Figure 2). Using a concatenation layer [18], outputs of the attention layer and decoder are combined and it produces the final target sequence.

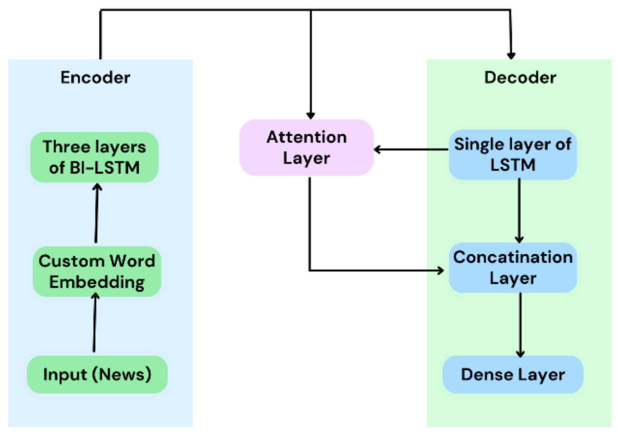


Fig. 2. Encoder-Decoder model

The news text tokens are sequentially inputted into the encoder and generate cell and hidden states. At every time unit (denoted by t in Figure 3), the decoder takes the previous word's word embeddings as input and maintains its decoder state. The attention distribution works as a probability distribution for the input words, guiding the decoder on what to concentrate on to generate subsequent words [16]. The vocabulary has been expanded to include new special tokens such as "EOS" and "SOS" [19]. (EOS and SOS stands for end of a sequence and start of a sequence, respectively.) The encoder processes an input text, denoted as $x = (x_1, x_2, ..., x_j)$, and converts it in the hidden states $h = (h_1, h_2, ..., h_t)$. Meanwhile, these hidden states are utilized by decoder to produce a summary $y = (y_1, y_2, ..., y_t)$.

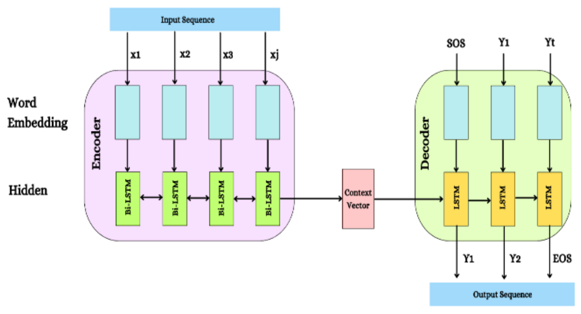


Fig. 3. Basic Seq2Seq Model

In this model, three Bi-LSTM layers are employed consecutively. Each layer processes the sequence of input in both directions (forward and reverse), generating sequences of hidden states for every time interval. The final layer's hidden and cell states from both directions are combined to create a context vector representing learned knowledge from the entire input data. This context vector serves as the base information for decoder. The decoder uses LSTM layer to generate sequences of outputs and states. Considering both encoder and decoder outputs, attention scores are calculated, which allows the system to concentrate on various segments of input, while producing each step of the output. The output from attention is then combined with the decoder's LSTM output to integrate essential information from the source sequence. To produce the final output for decoder, a time-distributed dense layer is applied to the concatenated output. The activation function used is softmax [27], indicating that the model undergoes training to generate probabilities for each element in the target vocabulary. This allows the model to provide a probability distribution of possible terms within the output sequence.

3.5.2. LSTM

It is a distinctive form of RNN, which stands out for its capability to manage sequential data effectively by overcoming the vanishing gradient problem. Unlike traditional RNNs, LSTMs excel at preserving long-range dependencies, making them well-matched for tasks involving sequences of information, such as summarization tasks [25].

In the classical architecture of an LSTM [23], three pivotal gates play a crucial role: input, forget, and output. These gates collaboratively manage the data flow inside the memory space. As indicated by c_{t-1} and c_t, the former embodies information stored in the previous memory cell, while the latter encapsulates the essence of the current memory cell. Simultaneously, h_{t-1} and ht represent the outcomes of the preceding and ongoing hidden states, where x_t serves as the input vector. The biases associated with these gates are denoted as b_f for the forget gate, b_i for the input gate, b_c for the cell state, and b_o for the output gate. The operation of bitwise multiplication is symbolized by X, and the addition operation is represented by +. Furthermore, the hyperbolic tangent function is denoted by tanh, and the sigmoid function is represented by σ.

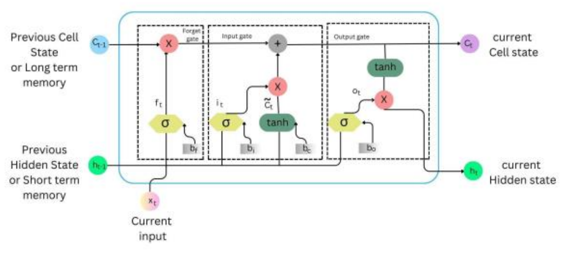


Fig. 4. LSTM unit architecture

(1) Forget gate

The forget gate determines the extent to which the memory content (c_{t-1}) will flow through the memory channel. It uses a sigmoid activation function applied to the sum of bias (b_f) and the weighted sum of x_t and h_{t-1} to produce a value between 0 and 1, indicating the contribution of the previous cell state to be retained.

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

(2) Input gate

The input gate takes charge of assessing the significance of the current input and making decisions of, till what extent this input must be preserved in the cell state for future reference. It calculates the potential memory content (c_t) by applying tanh function to sum of the bias (b_c) and the weighted sum of x_t and c_{t-1}. Employing a sigmoid activation function, it generates values ranging from 0 to 1 representing the impact of x_t and c_{t-1} on the current memory content (c_t).

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

$$c_t = (f_t\,c_{t-1} + i_t\,tanh(W_{xc}\,x_t + W_{hc}\,x_{t-1} + b_c) \qquad (3)$$

(3) Output gate

The output gate acts as a supervisor, deciding how information should travel to the hidden state from the cell state to regulate the entire information flow. The sigmoid function applied to the sum of bias (b_o) and the weighted sum of x_t and h_{t-1} produces output between 0 and 1, and the tanh function applied to current cell condition and output of sigmoid function produces the present hidden state (h_t).

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

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

3.5.3. Bi-LSTM

Bi-LSTM permits data to flow in both directions, capturing both past and future information. Unlike regular LSTM, which processes data in a single direction, also it enhances the model's ability to retain a full understanding of the input [25]. The bidirectional encoder captures comprehensive Information that considers what happened before and what might happen in the future. So Bi-LSTM is utilized in encoder, coupled with a unidirectional LSTM in the decoder. This strategic choice, elaborated in the preceding information, aims to optimize the model for abstractive text summarization. Context vector generated by bidirectional encoder is then utilized by the unidirectional decoder to generate coherent and contextually informed summary. This combination strikes a balance between capturing global context and maintaining sequential coherence, enhancing the summarization process.

3.6. Attention

The benefits of using this mechanism in the Seq2Seq model are:

1. Useful to handle variable length inputs and outputs.
2. It focuses on important part input i.e. key information of input sequence is maintained.
3. Helps in reducing redundancy in the generated output.

The proposed system incorporates the Bahdanau Attention Mechanism [5].

In this model, when an input sequence is passed through the encoder, at the last cell of encoder a context vector is formed that is acquired from the whole input sequence which is forwarded to the decoder at the start as one of the inputs. In the Encoder and Decoder framework, encoder processes input $x = (x_1, x_2, \ldots, x_{T_x})$ and forms a sequence of vectors into a context vector $c = q\,(\{h_1, h_2, \ldots, h_{T_x}\})$. Where $h_t = f\,(x_1, h_{t-1})$, $h_t \in R^n$ is a hidden state at time t, current time step's hidden state is denoted as h, while hidden states is used to formulate the context vector c. Functions f and q represent certain non-linear operations in this context. T_x is the length of the input source [5].

The decoder starts all its work with the help of the context vector. Even though LSTM can handle long sequences but when the input sequences are too long it may happen that during the formation of context vector, some of the words or sequences are lost, due to this the meaning of the input sequence can change and hence the generated target sequence is also irrelevant to the input i.e. incorrect output is generated to address above issue attention mechanism is used.

An attention layer is added in the encoder-decoder network. It helps the decoder to see the importance of each input sequence along with the context vector; this helps in preserving each sequence and avoid loss of information. Attention adds a bunch of new paths to the decoder from the encoder one per input so that each step of the decoder can access input value. It tries to get multiple encoded vectors from the source input [26]. Attention works as follows:

1. Calculates attention scores (e_{ij}) for each position j in the input sequence which depends on the current state of decoder s_i and hidden states from the encoder h_j.

$$e_{ij} = a(s_i, h_j) \quad (6)$$

here a is a scoring function that evaluates the compatibility among the encoder hidden state and decoder state

2. Softmax function[27] is employed on the attention scores to get normalized attention weights a_{ij}.

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (7)$$

where T_x is input sequence length.

3. The attention weights got from the softmax operation are further utilized to find the weighted sum of the input words. Weighted sum, often named as context vector (c_i), is then integrated with the current state of the decoder.

$$c_i = \sum_{j=1}^{T_x} a_{ij} . h_j \quad (8)$$

4. In this way the decoder cell gets access to each input information, no important information is lost and hence a better target is produced.

3.7. Evaluation

For assessing the performance of the model presented in this paper, ROUGE [7], [11] is utilized. It is a set of metrics primarily used to assess summaries generated by machines. It is an intrinsic measure that focuses on precision, recall, and F-score. To compute ROUGE scores, a summary predicted by a machine is compared against human formed summary [22]. The main components of ROUGE include: (i) ROUGE-N, assessing the overlap of N-grams (e.g., ROUGE-1 for overlap of unigrams, ROUGE-2 for overlap of bigrams, etc.), (ii) ROUGE-L, focused on the longest common subset, and (iii) ROUGE-S, utilizing a co-occurrence statistic based on skip-bigrams. In this study, the evaluation of summaries and the assessment of the model are conducted using ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-N represents an n-gram recall. N-gram is contiguous sequence of n items within a text. [14]. Formula for ROUGE-N:

$$ROUGE - N = \frac{No.\ of\ overlapping\ n - grams\ in\ P\ \&\ O}{Total\ No.\ of\ n - grams\ in\ O}$$

where $P = Predicted\ Summary$

$O = Original\ Summary$

4. Results and Discussion

4.1. Latent dimension analysis

The latent dimension is the hidden unit's count or neurons within the LSTM layer, influencing the dimensionality of the internal representations or memory cells that the LSTM layer can learn during training.

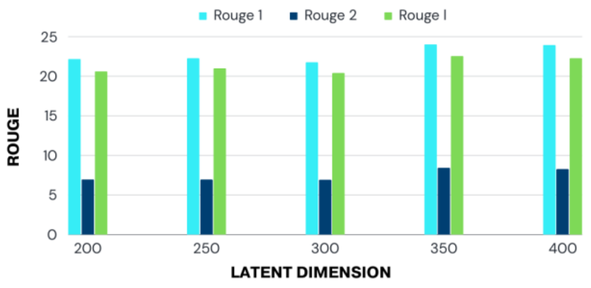


Fig. 5. Latent Dimension Analysis

Upon analysing ROUGE values across various latent dimensions, it was determined that setting the latent dimension to 350 produces the most favourable results, which is concluded from figure 5.

4.2. Early stopping

This study involved training the model with training and validation data for 60 epochs (maximum limit) with early stopping. Early stopping was implemented by monitoring validation loss, ensuring the neural network stopped training at the right time to prevent overfitting. Batch size is set as 32. sparse categorical cross-entropy loss function is utilized to compute training and validation loss and to address potential memory issues [21].

4.3. Input coverage impact on ROUGE metrics

Table 2. Comparison of ROUGE values using average length of summary and input news text and maximum length of text and summary.

Length Coverage	ROUGE-1	ROUGE-2	ROUGE-L
Max	24.04	8.43	22.57
Avg	16.28	4.46	15.09

Table 2 illustrates that the ROUGE metric for maximum length surpasses that for average length. It is also observed that the model generates better summaries for maximum length. For the "max" type, a fixed length of 30 words for text and 15 words for summaries is employed. On the other hand, for the "avg" type, a fixed length of 19 words for text and 10 words for summaries is set. The model's effectiveness is examined while considering both maximum length and average length of text and summaries in the dataset. When taking into account the maximum length, 99% of instances are taken without truncating. However, when considering the average length, the coverage drops to 58%. In other words, only 58% of instances are taken without truncating, while the remaining 42% require truncation. Our findings suggest that the model exhibits more effective performance when evaluated based on the maximum length, which covers the maximum input data, as opposed to the average length.

4.4. Comparison of optimizers

Table 3. Comparison of ROUGE values using Adam optimizer and Rmsprop.

Optimizer	es	ROUGE-1	ROUGE-2	ROUGE-L
Adam	4	24.04	8.43	22.57
Rmsprop	9	24.28	7.86	22.80

In this study, both Rmsprop [19] and Adam optimizers [23] were employed, and Table 3 outlines ROUGE values and ‘es’ represents the number of epoch required for training model with early stopping for both optimizers. Notably, Adam halted training at the 4th epoch, while Rmsprop continued until the 9th epoch. Despite a marginal difference in rogue scores between the two optimizers, with only a single-point variation, it is noteworthy that the training duration for the model was twice as long with RMSprop compared to Adam.

4.5. Learning rate analysis

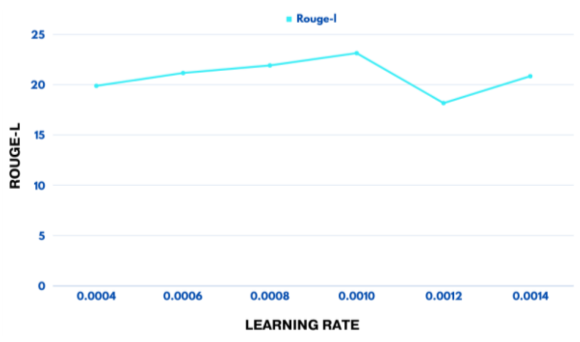


Fig. 6. Learning Rate Analysis

Figure 6 illustrates the relationship between the learning rate and ROUGE-l values. The learning rate holds significant importance as one of the key hyperparameters that dictate the step size for adjusting a model's weights during training, impacting both convergence speed and stability. Notably, at the default learning rate value is 0.001 for the Adam optimizer, we observe high ROUGE values. Specifically, Rogue-1 is 24.27, ROUGE-2 is 8.57, and ROUGE-l is 23.13. Based on these results, the decision was made to stick with the default learning rate, as it consistently yields the maximum ROUGE value.

4.6. Unidirectional and bidirectional LSTM encoders

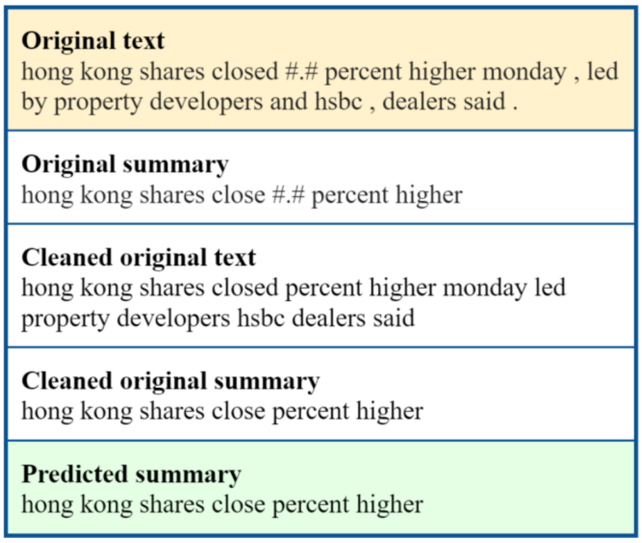
Table 4. ROUGE values for unidirectional and bidirectional LSTM encoder model

Encoder type	ROUGE-1	ROUGE-2	ROUGE-L
Bi-LSTM	24.04	8.43	22.57
LSTM	25.64	8.77	24.05

From table 4, We notice that the Rouge1 and RougeL scores of the unidirectional LSTM encoder model are slightly higher compared to the Bi-LSTM model. However, the Rouge2 score is greater for the model with a Bi-LSTM encoder. Summaries generated by model using Bi-LSTM are more readable than unidirectional LSTM encoder model.

4.7. Summary generated by our model

The samples below (Figure 4, Figure 5, Figure 6) display the outputs obtained by our model. In these figures, "original text" refers to the text from the dataset, "original summary" corresponds to the summary present in the dataset, "cleaned text" and "cleaned summary" represent the text and summary obtained through the data cleaning process in the text preprocessing step, and "predicted summary" refers to summary produced by proposed model.



Original text
hong kong shares closed #.# percent higher monday , led by property developers and hsbc , dealers said .

Original summary
hong kong shares close #.# percent higher

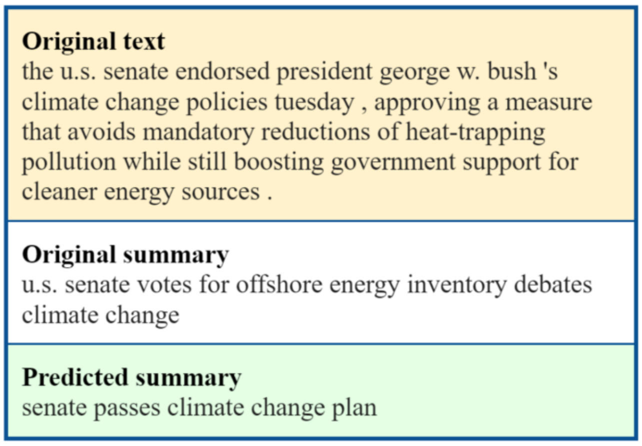
Cleaned original text
hong kong shares closed percent higher monday led property developers hsbc dealers said

Cleaned original summary
hong kong shares close percent higher

Predicted summary
hong kong shares close percent higher

Fig. 7. Comparison of Original and Predicted Summaries: Sample 1 [6]

In Figure 7, it is evident that the predicted summary matches the cleaned original summary.

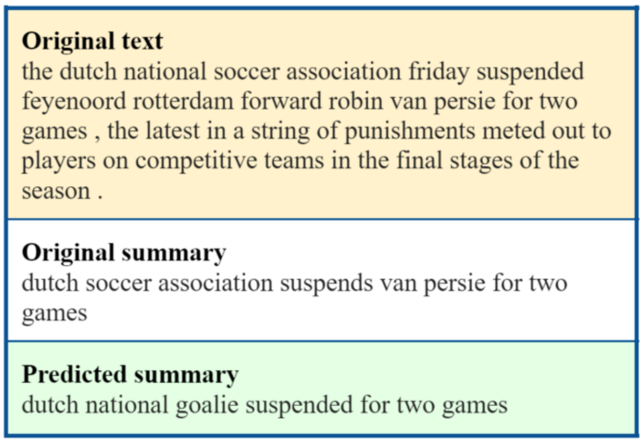

Original text
the u.s. senate endorsed president george w. bush 's climate change policies tuesday , approving a measure that avoids mandatory reductions of heat-trapping pollution while still boosting government support for cleaner energy sources .

Original summary
u.s. senate votes for offshore energy inventory debates climate change

Predicted summary
senate passes climate change plan

Fig. 8. Comparison of Original and Predicted Summaries: Sample 2 [6]

In Figure 8, the predicted summary is better than original summary. Also certain words in the predicted summary are not in original text.


Original text
the dutch national soccer association friday suspended feyenoord rotterdam forward robin van persie for two games , the latest in a string of punishments meted out to players on competitive teams in the final stages of the season .

Original summary
dutch soccer association suspends van persie for two games

Predicted summary
dutch national goalie suspended for two games

Fig. 9. Comparison of Original and Predicted Summaries: Sample 3 [6]

In Figure 9, the predicted summary is not the same as the original, however it is relevant considering the original text.

4.8. Discussion

Our model demonstrates varying levels of performance in summary prediction. It accurately predicts the summary as the original in some cases, surpasses the original summary in others, and in a few instances, generates irrelevant summaries. The model tends to produce more accurate and relevant summaries when the input news closely aligns with the patterns observed in the training dataset such as domain of news like news related to currency fluctuations. However, in certain cases, it may produce irrelevant output.

The hardware requirements for this study are as follows:

1. The minimum RAM requirement for training and testing the model is 12 GB, a recommended amount of 32 GB or more.

2. Google Colab's T4 GPU processor is utilized for optimal performance. While the CPU can also be used, however it necessitates more than twice the time compared to the T4 GPU when utilized as the runtime.

5. Conclusion

In this study, an approach for single-line abstractive summarization, specifically for headline generation, is proposed. The proposed system uses an encoder-decoder architecture which incorporates attention mechanism, designed for English news. The usage of Bi-LSTM in the encoder produces a more readable summary than unidirectional LSTM. Our findings indicate that incorporating the attention mechanism, along with Bi-LSTM encoding and LSTM decoding, results in improved summarization outcomes. Based on our experimental analysis, a model is configured with an embedding dimension of 300, and a latent dimension of 350, and employed an Adam optimizer having 0.001 learning rate. Additionally, early stopping was implemented to mitigate overfitting. The peak length of both the summary and text covered approximately 99% of the input. Our model achieved its optimal performance with ROUGE-1, ROUGE-2, and ROUGE-L scores of 24.27, 8.57, and 23.13, respectively. The model displays diverse performance in summary generation, ranging from accurately predicting the original to surpassing it or occasionally producing unrelated summaries.

Certain limitations were recognized in this experiment. An essential constraint involves the necessity to predefine the text and summary lengths to a fixed number before feeding the data within the neural network. Model's training and testing phases demand a considerable time frame and hardware resources that exceed the conventional capacities of personal computers or laptops.

In the future, our objective is to assess the model's effectiveness across a range of datasets and further enhance its capabilities by training it on larger datasets and additional datasets for news summarization. In addition, future work will explore the potential of extending the model to domains beyond news text and headlines, with a particular focus on handling multi-sentence summarization. Also try extending the model to different languages.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Suleiman D, Awajan A. Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges. Mathematical problems in engineering. 2020 Aug 24;2020:1-29.

[2] Dilawari A, Khan MU, Saleem S, Shaikh FS. Neural Attention Model for Abstractive Text Summarization Using Linguistic Feature Space. IEEE Access. 2023 Feb 27;11:23557-64.

[3] Jobson E, Gutiérrez A. Abstractive text summarization using attentive sequence-to-sequence rnns.

[4] Rush AM, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685. 2015 Sep 2.

[5] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. 2014 Sep 1.

[6] Parker, Robert, et al. English Gigaword Fifth Edition LDC2011T07. Web Download. Philadelphia: Linguistic Data Consortium, 2011.

[7] Resnik P, Niv M, Nossal M, Schnitzer G, Stoner J, Kapit A, Toren R. Using intrinsic and extrinsic metrics to evaluate accuracy and facilitation in computer-assisted coding. InPerspectives in health information management computer assisted coding conference proceedings 2006 Aug (pp. 2006-2006).

[8] Liu F, Liu Y. Exploring correlation between ROUGE and human evaluation on meeting summaries. IEEE Transactions on Audio, Speech, and Language Processing. 2009 Jun 10;18(1):187-96.

[9] Jiang J, Zhang H, Dai C, Zhao Q, Feng H, Ji Z, Ganchev I. Enhancements of attention-based bidirectional lstm for hybrid automatic text summarization. IEEE Access. 2021 Sep 3;9:123660-71.

[10] Li Z, Peng Z, Tang S, Zhang C, Ma H. Text summarization method based on double attention pointer network. IEEE Access. 2020 Jan 10;8:11279-88.

[11] Hanunggul PM, Suyanto S. The impact of local attention in lstm for abstractive text summarization. In2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) 2019 Dec 5 (pp. 54-57). IEEE.

[12] Nallapati R, Zhou B, Gulcehre C, Xiang B. Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023. 2016 Feb 19.

[13] Wong KF, Wu M, Li W. Extractive summarization using supervised and semi-supervised learning. InProceedings of the 22nd international conference on computational linguistics (Coling 2008) 2008 Aug (pp. 985-992).

[14] Moratanch N, Chitrakala S. A survey on extractive text summarization. In2017 international conference on computer, communication and signal processing (ICCCSP) 2017 Jan 10 (pp. 1-6). IEEE.

[15] Mridha MF, Lima AA, Nur K, Das SC, Hasan M, Kabir MM. A survey of automatic text summarization: Progress, process and challenges. IEEE Access. 2021 Nov 22;9:156043-70.

[16] See A, Liu PJ, Manning CD. Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368. 2017 Apr 14.

[17] Shi T, Keneshloo Y, Ramakrishnan N, Reddy CK. Neural abstractive text summarization with sequence-to-sequence models. ACM Transactions on Data Science. 2021 Jan 3;2(1):1-37.

[18] Sakhare DY. A Sequence-to-Sequence Text Summarization Using Long Short-Term Memory Based Neural Approach. International Journal of Intelligent Engineering & Systems. 2023 Mar 1;16(2).

[19] Singh S, Singh JP, Deepak A. Deep Learning based Abstractive Summarization for English Language. InWorking Notes of FIRE 2022-Forum for Information Retrieval Evaluation, Kolkata, India 2022 Dec 9.

[20] Kouris P, Alexandridis G, Stafylopatis A. Abstractive text summarization: Enhancing sequence-to-sequence models using word sense disambiguation and semantic content generalization. Computational Linguistics. 2021 Dec 23;47(4):813-59.

[21] Wazery YM, Saleh ME, Alharbi A, Ali AA. Abstractive Arabic text summarization based on deep learning. Computational Intelligence and Neuroscience. 2022 Jan 11;2022.

[22] Siddiqui T, Shamsi JA. Generating abstractive summaries using sequence to sequence attention model. In2018 International Conference on Frontiers of Information Technology (FIT) 2018 Dec 17 (pp. 212-217). IEEE.

[23] Masum AK, Abujar S, Talukder MA, Rabby AS, Hossain SA. Abstractive method of text summarization with sequence to sequence RNNs. In2019 10th international conference on computing, communication and networking technologies (ICCCNT) 2019 Jul 6 (pp. 1-5). IEEE.

[24] Rahman MM, Siddiqui FH. An optimized abstractive text summarization model using peephole convolutional LSTM. Symmetry. 2019 Oct 14;11(10):1290.

[25] Anvitha Aravinda, Gururaja H S, Padmanabha J, Unique Combinations of LSTM for Text

Summarization, International Journal of Engineering Research & Technology (IJERT) ICEI – 2022 (Volume 10 – Issue 11)

[26] Sanjabi N. Abstractive text summarization with attention-based mechanism (Master's thesis, Universitat Politècnica de Catalunya).

[27] Huang L, Wu H, Gao Q, Liu G. Attention Localness in Shared Encoder-Decoder Model For Text Summarization. InICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2023 Jun 4 (pp. 1-5). IEEE.

[28] Parker, Robert, et al. English Gigaword Fifth Edition LDC2011T07. Web Download. Philadelphia: Linguistic Data Consortium, 2011.