Architecture
Image Studies

Vol.7, Issue 1, pp.2914-2927, 2025
DOI: https://doi.org/10.62754/ais.v7i1.1342

https://journals.ap2.pt/index.php/ais/index

Abstractive Text Summarization using LSTM, GLOVE, and TensorFlow

Neeraj Kumar Sirohi[1], Mamta Bansal[2]

Abstract

Automatic Text summarization (ATS) is a fundamental task in NLP (natural-language-processing), aimed at generating concise and coherent summaries of longer texts. In this study, we investigate the use of deep-learning techniques, with a focus on LSTM (Long Short-Term Memory) neural networks, Global Vectors used in Word Representation (GLOVE) embeddings, and the TensorFlow framework, for abstractive text summarization.Our approach leverages the power of LSTM networks to capture sequential dependencies in the input text, enabling the generation of abstractions that extend beyond simple sentence extraction. GLOVE embeddings are used to denote the words in a constant vector space, thereby improving the model's comprehension of semantic relationships between words. TensorFlow provides the computational framework for efficient model training and deployment.We perform experiments on diverse datasets to assess the execution of our abstractive text summarization model. Our findings demonstrate that integrating LSTM, GloVe embeddings, and TensorFlow significantly enhances the quality and fluency of generated summaries compared to traditional extractive summarization approaches.This study contributes to the development of abstractive text summarization techniques, offering a promising approach to distill important details from textual data automatically. Furthermore, the use of open-source tools like TensorFlow makes our model accessible and adaptable for many different applications in the field of natural language processing.

Keywords: *Abstractive Summarization, LSTM, Glove, TensorFlow, Natural Language Processing, Text Summarization.*

Introduction

In the ever-expanding realm of data, the capability to extract significant sizes of articles into brief and important summaries has become increasingly crucial. One difficult issue in natural language processing is text summarizing, which is the extraction of essential information from a given document while preserving its core meaning. This process finds applications in various domains, from news articles and research papers to social media posts. Before proceeding to our research about automatic text summarization first, we understand “what summary means. Various researchers provide different definitions of summary but best defined by[1] " a text that is shorter than half of the original texts' length, based on one or more source texts, and contains the most pertinent material from those sources.”. Other definitions of the summary are as follows

- According to [2] “Summarization is the method which works with enormous information and taking out relevant information by concentrating solely on the crucial details”.
- Hove (2005) “A summary refers to information that has been extracted from single or multiple texts and includes a significant amount of the content found in the source text or texts, and its length does not exceed half of the original text(s)”.
- Fattah and Ren (2008) said “Text summarization involves the automated generation of a condensed version of a given text, aiming to deliver valuable information for a specific purpose”.

[1] Research Scholar, Shobhit Institute of Engineering & Technology, Meerut, India, Email:ID: neeraj.2019060045@shobhituniversity.ac.in , Email:ID: neerajsiroh@gmail.com, (Corresponding Author)
[2] Shobhit Institute of Engineering & Technology, Meerut, India, Email: ID: mamta.bansal@shobhituniversity.ac.in

Generally, we can say. A summary is a brief and condensed representation of the main points, essential elements, or key ideas of a longer piece of text, speech, or other form of communication. It aims to provide a concise overview of the original content, allowing readers or listeners to quickly grasp the central themes without going through the complete text. The purpose of a summary is to offer a quick understanding of the primary content, making it useful for readers who want a quick overview or for those who need to grasp the essential information efficiently. When the summary is created automatically, it is called Automatic text summarization, a critical NLP (natural language processing) area . It refers to the process of using computational methods, algorithms, or artificial intelligence techniques to generate summaries of textual content without human intervention. ATS (Automatic-text-summarization) can be achieved by either extractive or abstractive techniques[3]. In the extractive method, essential sentences or phrases are selected from the primary document. This technique has two major challenges. The first challenge involves effectively extracting the most pertinent information and key principles from the source document. This process requires identifying and selecting the content that is crucial for conveying the main ideas of the text. The second challenge is to organize and present this extracted information coherently and logically, creating a summary that accurately reflects the essential elements of the original document. The abstractive procedure involves rephrasing key sections and fundamental concepts of a text document to identify them. The basic steps and structure of the automatic text summarization process (extractive or abstractive) are as follows.

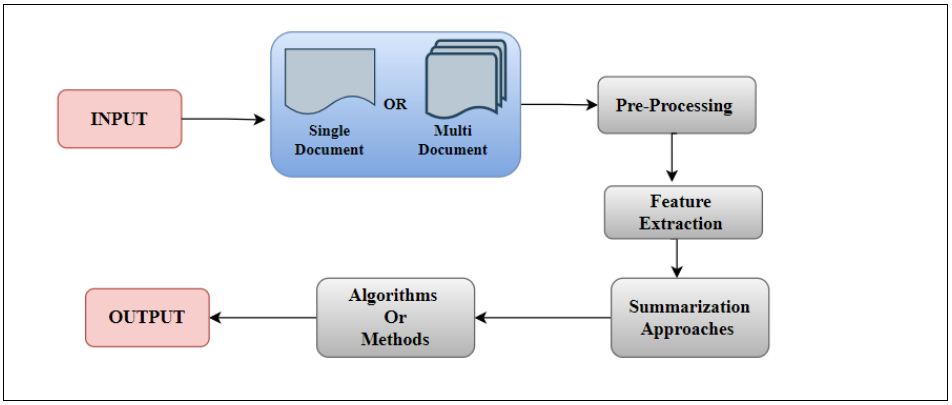

Figure 1: Complete process and steps involved in ATS

Pre-processing:- During the pre-processing stage, semantic methods are utilized to prepare input text documents. These techniques cover fundamental operations like stemming, which reduces words to their common root form, stop word removal, sentence segmentation, and punctuation mark removal.

Feature Extraction: Sentences are extracted crucially to the summarization process, involving the selection of various features from the Original text. During the feature extraction stage, these attributes are applied to each phrase, and the sentences that receive the best scores are selected for the summary.

Summarization Methods: The initial and most important step in the summarization strategy is identifying effective approaches. Several approaches focus on picking up the best keywords and sections from the literature, while still others include reducing the primary content and rearranging sentences.

Algorithms: Algorithms or techniques provide a more precise technique to describing text summarization. Various techniques and algorithms are employed under different tactics to produce an improved form of the summarized document

Developing an abstractive summary model is capable of generating concise summaries with a deep understanding of domain knowledge. In this paper, we focused on LSTM networks and GloVe embeddings with TensorFlow to generate an abstractive text summarization model. Although we use the Amazon Food Review dataset as a reference, the dataset is not the main focus of our study but rather a means to determine how well our model works. Our goal is to show how deep learning methods—specifically, LSTM-based architectures—can improve abstractive summarization quality on many different kinds of textual datasets

An optimized LSTM-based summarizing model, a study of the effect of word embeddings on summary quality, and an assessment of the effectiveness of the model using common summarization metrics are the main contributions of this study. This effort intends to enhance the field of AI-driven text production and comprehension by reorienting the attention from the dataset itself to the methodological developments in abstractive summarization.

Related Work…

Generally, Automatic text summarization (ATS) can be categorized into two categories (a) Extractive text summarization and (b) Abstractive text summarization.

Extractive Text- Summarization:- The summary can be generated by selecting the key sentences and phrases from the original text without modifying them. A lot of research is done on extractive text summarization table below describes a comparative study on different methods

Table 1: Comparative studies of Extractive text summarization approaches

Approaches	Key Techniques	Advantages	Limitations	Models
Statistical Methods	TF-IDF, Word Frequency, Position-Based	Simple, Fast, Language-Independent	Lack of deep contextual understanding	Luhn (1958)[4], Edmundson (1969)[5]
Graph-Based Approaches	TextRank, LexRank, Sentence Similarity	Captures relationships between sentences	Computationally expensive for large texts	TextRank (2004), LexRank (2004)[6]
Machine Learning-Based	Naïve Bayes, SVM, LSTMs	Learns patterns from labelled data	Requires training data, domain-dependent	Nallapati et al. (2017)[7], Cheng & Lapata (2016)
Transformer-Based Models	BERT, RoBERTa, Pegasus	High accuracy, Context-aware, Handles long texts	Computationally expensive	BERTSUM (2019)[8], MatchSum (2020)

Extractive text summarization has many limitations. some of the major limitations discussed by various researchers are that in this technique, sentences are simply picked from the original text without modification, which is results in a lack of smooth logical flow, and the summary may also lack connectors, transitions make summary difficult to read [9].

The second limitation is that extractive methods sometimes select sentences that convey the same information, which leads the redundancy [10] proved that extractive methods collapse to efficiently handle the redundancy issue

One more major issue with the extractive method is that it fails to handle longs and multiple documents due to sentence selection limits[8]

Abstractive text summarization:- On the other hand, there is minimal research on abstractive methods. Deep learning finds extensive applications in abstractive-text summarization, where an abstractive method of summarization focused on recurrent-neural-networks proves effective for generating summaries[11], [12] suggest that Deep learning finds extensive applications in abstractive-text summarization (ATS). Recurrent neural networks (RNN)-based abstractive text summarization approach generates a summary with a higher ROUGE score.. Another ATS model, employing a sequence-to-sequence approach, was explained [13]. This model utilizes a Gated-Recurrent Unit (GRU) along with a bidirectional neural network for its construction in 2019 [14]. They proposed an advanced abstractive summarization method for biomedical text, employing two primary frameworks. These are basic approaches of neural networks based on the sequence-to-sequence attention basis and the pointer-generator outline. A one-layer (RNN) unidirectional (linear) recurrent neural network and (GRU) gated-recurrent units for the encoder and decoder are used in the sequence-to-sequence attentional architecture. Their approach is evaluated on the MEDLINE dataset using metrics such as ROUGE, UMLS, MeSH, and TF-IDF word appearance. For R-1, R-2, and R-L, the system's ROUGE metrics yielded values of (42.43, 21.59, and 36.89), respectively.

One year later [15] They introduced a system designed for extractive text summarization through the application of a recurrent neural network. The planned technique comprises three stages: sentence

encrypting, word ranking, and summary creation. Coreference resolution is employed to enhance the system's performance.

In 2021 [16] proposed an abstractive text summarizing model based on deep learning called T5, PropheNet, and BART, and used it to create an abstractive summarization model specifically designed for summarizing technical literature topics. In their approach, deep learning models are employed to generate headlines from the abstracts of scientific papers. The scheme's performance is then assessed through ROUGE metrics. The results indicate that the fine-tuned ProphetNet excels in f-1 measures and ROUGE while the fine-tuned T5 method demonstrates a strong ROUGE precision score.

An adapted form of the conventional RNN, known as a convolutional RNN, demonstrates effectiveness in various prediction and classification tasks. Specifically, for improving nowcasting predictions, a model employing a convolutional RNN (ConvRNN) was created by extending the fully connected RNN [13]. This ConvRNN model takes into account the entire correlation within the data, resulting in more accurate nowcasting predictions compared to traditional RNN-based models. Furthermore, a convolutional RNN has found application in time series classification. Then, a model that is a fully convolutional LSTM (long short-term memory), which demonstrates highly effective classification of time series [17]. The design of the convolutional RNN architecture enables the model to consider the complete time stamp dependency during training. Given the effective performance of convolutional RNN in diverse groupings and forecasts, it is anticipated to excel in abstractive text summarization as well. The ConvRNN used in abstractive text summarization has some limitations as it restricts access to the preceding cell state if the output gate is locked.

The discussed abstractive methods also have some limitations. A neural network model sometimes produces false and misleading information that is not present in the source document [18] found that approx. 30%-40% of abstractive summaries contain hallucinated information. Abstractive models need a large amount of training and computational data, which requires more time to execute[8] Noted that fine-tuning the training model needs a significant GPU to execute

We have introduced ATS model by using LSTM along with TesorFlow and Glove for word embedding. Multiple sequential layers comprise the proposed model: the dense layer, word-embedding layer, LSTM, and data entry (Input) layer. In this setup, the embedding-layer receives word vectors generated by GloVe model after transforming the order of texts into word vectors. Subsequently, the embedding layer passes the resulting fixed-length vector-encoded statements to the LSTM layer. The LSTM layer uses three gates and a cell of memory to start the computing process for each input within a single cell. then forwards the dense layer’s outcomes to the system's next layer. The system modifies its parameters via training and validation procedures, finally producing a predicted summary.

Transformer-oriented structures, which utilize self-attention processes to better describe long-range dependencies than recurrent networks, have dominated recent developments in abstractive text summarization. On common summarization benchmarks, models like BART, T5, and PEGASUS have shown state-of-the-art performance.

Although Transformer-based models produce better summaries, they usually need a lot of pretraining, a lot of processing power, and a lot of memory overhead, which may limit their use in contexts with limited resources

Experimental Setup

Dataset

There are mainly three datasets used for this problem :

Amazon Product Reviews (which we will use in our case)

Daily Mail corpus

CNN/Daily News dataset

A prominent benchmarking dataset for abstractive text summarization, particularly for neural sequence-to-sequence and deep learning-based methods, is the CNN/DailyMail dataset. It is made up of human-written multi-sentence summaries of news articles gathered from the CNN and Daily Mail news websites.

Each sample in the dataset contains:

a lengthy news story (document) that is usually several hundred words long.

a matching reference summary that summarizes the article's main points and is authored by qualified editors using bullet points.

The dataset is ideal for abstractive summarization since the summaries need semantic comprehension, content compression, paraphrasing, and abstraction rather than just extracting sentences from the source text. Because of this, the dataset serves as a suitable baseline for assessing how well neural models can provide logical and insightful summaries.

The suggested LSTM-derived abstractive summarization technique used in this study is trained and assessed using the CNN/DailyMail dataset. Both texts and summaries undergo common preparation procedures like sentence normalization, tokenization, noisy symbol removal, and length reduction before training. To make sequence-to-sequence learning easier, unique tokens that indicate the beginning and end of sequences are included.

In accordance with conventional procedure, the dataset is divided into training, validation, and test sets to ensure impartial assessment and comparison with current techniques. The conceptual soundness, robustness, and comparability of the suggested method with cutting-edge abstractive summarization methods documented in the literature are guaranteed by the usage of the CNN/DailyMail dataset.standard for assessing neural models' capacity to provide logical and instructive summaries.

Data Pre-processing : [19] [20]

Text pre-processing is the crucial "cleaning" step in Natural Language Processing (NLP) that converts unstructured human language into a format that computers can effectively analyze.

"At this stage, the collected data undergoes processing to organize it for training the model. This phase is divided into numerous steps, as outlined below:

Data cleaning: Unnecessary data is removed during this step.

Tokenization: The text is divided into individual units called tokens, each represented by a specific number.

Padding: Sequences generated in the previous step are adjusted to a uniform length. This involves pre-padding sequences based on the maximum sequence length in the dataset [21].

Noise Removal: The primary phase of defense is noise reduction. It entails removing any formatting or letters that don't contribute much for our particular goal.

What it removes:- It eliminates excess whitespace, special characters (such as

punctuation, hashtags, and emojis), and HTML tags (such as or <p>)

For example :- we have an input "Look at this link <a href='…..'> Click this link: #NLP 2024

Output:- "Check out this link Click here NLP 2024".

Stop-Word Removal :- The most prevalent words in a language are stop-words, which offer grammatical structure but have relatively little distinctive meaning or "flavor."

"The," "is," "at," "which," "on," and "and" are frequently used stop words.

Example:-: "The cat is sitting on the mat."

Output:- "Cat sitting mat"

Model Configuration

For abstractive text summarization, the suggested model uses a sequence-to-sequence (Seq2Seq) encoder–decoder architecture constructed using Long Short-Term Memory (LSTM) networks. Considering an input document

The goal is to create a summary S={s1,t2,…,tN} where M≪N, such that the summary captures the essential semantic information of the input text. T ={t1,t2,…,tN} consists of N words taken from a vocabulary V V. Recurrent neural networks are used to solve this job, which is characterized as maximizing the conditional probability $P(S*T)$P(S∗T).

The suggested architecture creates summaries word-by-word in an autoregressive fashion, in contrast to classification-based models that forecast a fixed output.

The phases of the proposed system are illustrated in the Figure 2 and Layer-wise architecture and parameter are describe in Table 2.

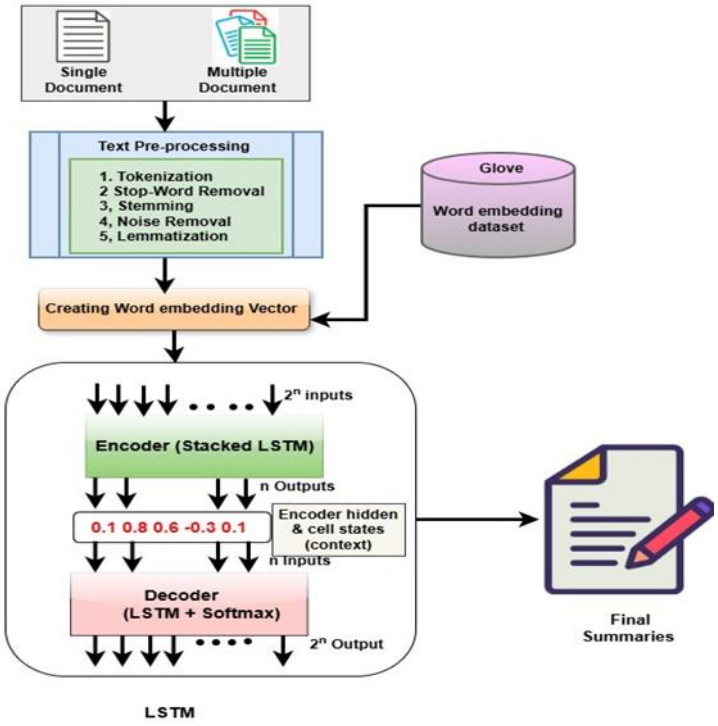

Figure 2: Architecture of the proposed LSTM-based abstractive summarization

Table-2 model Layer-wise architecture and parameter breakdown of the proposed LSTM-based encoder–decoder abstractive summarization model with attention.

No	Layer Name	Layer type	Output shape	Trainable Parameters	Description
1	Input_1	Input layer	(none,80)	0	Accepts the tokenized input document sequence of fixed length 80
2	Embedding	Embedding	(None, 80, 500)	10,792,000	Maps input tokens to 500-dimensional GloVe-based word embeddings
3	LSTM	Encoder LSTM (Layer 1)	(None, 80, 500)	2,002,000	First encoder LSTM layer extracting contextual representations
4	Input_2	Input layer	(None,None)	0	Accepts tokenized target summary sequence for decoding
5	LSTM_1	Encoder LSTM (Layer 2)	(None, 80, 500)	2,002,000	Second stacked encoder LSTM layer refining representations
6	Embedding_1	Embedding	(None, None, 500)	3,078,500	Embedding layer for decoder input tokens
7	LSTM_2	Encoder LSTM (Layer 3)	(None, 80, 500)	2,002,000	Third encoder LSTM layer producing final encoder states
8	LSTM_3	Decoder LSTM	(None, 80, 500)	2,002,000	Decoder LSTM initialized with encoder final states
9	Attention_Layer	Attention	(None, None, 500)	500,500	Computes context vectors by aligning encoder and decoder states
10	Concat_Layer	Concatenation	(None, None, 1000)	0	Concatenates decoder output and attention context vectors
11	TimeDistributed	Dense + Softmax	(None, None, 6157)	6,163,157	Generates probability distribution over the target vocabulary
Total Trainable Parameters				**28,542,157**	

Layer-wise Explanation of the Proposed Model

The suggested abstractive summarization approach uses layered LSTM networks to build a sequence-to-sequence encoder–decoder framework with a mechanism for attention.

After an embedding layer converts continuous words into 500-dimensional word arrays, the encoder gets the input text chain. Three layered LSTM layers with 500 hidden units each come next, gradually extracting situational and semantic data from the input material. The decoder receives the encoder's final hidden and cell states in order to generate a summary.

One LSTM level with 500 secret units and a special embedding layer make up the decoder. based on previously created terms and encoder context, the decoder produces the summary chain in an adaptive fashion.

An attention method is used to enhance flow of information and overcome the drawbacks of limited-length context structures.

For every decoding phase, the attention layer produces a dynamic context vector by calculating match scores among encoder outputs and decoder states. A stronger description is created by concatenating this context vector with the decoder output.

Lastly, the combined representation is mapped to a probability distribution across the target vocabulary of size 6,157 at each time step by a TimeDistributed Dense layer with softmax activation.

The embedding layers, stacking LSTM encoder stages, decoder LSTM, attention mechanism, and output visualization level all contribute to the model's 28,542,157 trainable parameters. The deep learning framework's model summary is used to automatically calculate this parameter count.

Encoder–Decoder LSTM Framework

Three overlapping LSTM layers make up the encoder, which successively processes the input document. A word embedding stage initially converts each input word into a dense vector version. The

encoder extracts essential information and long-range dependencies from the input text. In order to start the summary generation task, the decoder receives the encoder's final hidden and cell states, which create a semantic illustration of the input document.

A single-layer LSTM network is used to construct the decoder. The decoder forecasts the subsequent word in the summary at each decoding stage based on:

The encoder's contextual states

The word that was previously created

The internal hidden state of the decoder

Abstractive summaries can be produced with ease and coherence thanks to this approach.

Embedding Layer[22] [23] [25]

Independent word tokens are transformed into compact numerical vectors that represent semantic links between words using an embedding layer. Pre-trained GloVe word embeddings are used to set up the embedding matrix in order to improve representation quality and speed up convergence.

LSTM Encoder

Memory cells controlled by forget, input, and output gates make up the LSTM encoder. These gates manage information flow and enable the network to store pertinent contextual information over extended periods. The model is better able to learn hierarchical representations of the input text thanks to the stacked encoder structure.

LSTM Decoder

A single token at a time, the decoder creates the summary order. Instructor forcing is implemented throughout training by providing the decoder with the genuine summary words. Until an end-of-sequence token appears or the maximum summary length has been met, the decoder uses previously obtained tokens during prediction.

Output Layer [24]

At every decoding stage, a fully linked Dense layer with softmax activation is used to calculate a probability distribution across the target vocabulary. The following word in the created summary is chosen to be the token with the highest probability.

Text Decoder: The decoder is responsible for generating each word in the output sequence, relying on two sources of information:

Context Vector: This is the primary text determined and delivered through the encoder.

Generated Sequence: This refers to the finishing order of text already considered part of the summary.

The context matrix in a basic encoder-decoder paradigm is usually a fixed-length encoding, although it may involve further computations if an attention mechanism is used. Specific training, such as the scattered representation of all created words via word embedding, improves the produced chronological sequence. As [18] explains with a diagram, X represents the key text, Enc stands for the encoder illustrating the gist representation of the key text article, and yc denotes the arrangement of earlier produced words.

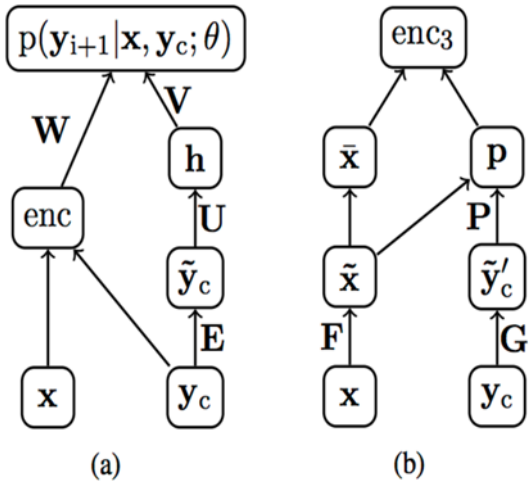

Figure 3 Text Summarizer Decoder

Training Details

The Adam optimizer, whose flexible learning rate makes it ideal for sequence learning problems, is used to train the suggested LSTM-based encoder–decoder model. A batch size of 64 is used for the training in order to balance model convergence with computational efficiency. To examine how training time affects convergence behavior,

the model is trained across ten and fifteen epochs. To avoid overfitting, early pausing

is implemented based on validation loss.

To guarantee impartial assessment and repeatability, every experiment is carried out under the same training conditions.

Evaluation Matrix

Since there may be more than one accurate summary for a given document, accuracy is not a good metric for assessing abstractive text summarization. Therefore, ROUGE-1, ROUGE-2, and ROUGE-L metrics—standard assessment measures for summarization tasks—are the only metrics used in this study to assess model performance. The main purpose of reporting training and validation losses is to examine convergence behavior during model optimization.

Result ad Discussion

This study employed the GloVe model for word embedding, which is focused on the Wikipedia 2014 and Gigaword5 datasets. The GloVe model's vector dimension is 100, and it generated a vocabulary of 29,519 words. The system uses one layer of LSTM in the decoder and three layers of LSTM in the encoder, each with a dimension of 300. There are 9,904,842 parameters in the system overall. 512 is the batch size, with a fixed dropout rate of 0.5. The Adam optimizer, which has a 0.001 learning rate, is employed to modify the parameters of the system.

Accuracy is not a suitable metric for evaluating abstractive text summarization, as multiple valid summaries may exist for a given document. Therefore, this study evaluates model performance exclusively using ROUGE-1, ROUGE-2, and ROUGE-L metrics, which are standard evaluation measures for summarization tasks.”

. TensorFlow was utilized to construct the neural network, with Jupyter Notebook.

Training Convergence Analysis

Training and validation loss results over epochs are recorded in Table 3 and shown in Figure 4 to examine the learning performance and consistency of the suggested model. As the number of epochs rises, training loss and validation loss both steadily decline, demonstrating the model's successful optimization and convergence.

The model appears to have minimal overfitting and to generalize quite well to new data based on the comparatively minor difference between training and validation loss.

In order to show convergence behavior during training, token-level training accuracy is only reported for diagnostic purposes. Nevertheless, accuracy is not regarded as a reliable assessment criterion for abstractive text summarization. As a result, only ROUGE-1, ROUGE-2, and ROUGE-L metrics—discussed below—are used to assess final model performance.

Table-3 Training and Validation Loss Across Epochs

Epoch	Training Loss	Validation Loss
1	7.1826	6.2277
3	6.4437	6.1188
5	6.3729	6.0490
8	6.2843	5.9815
10	6.2276	5.9400
12	6.1685	5.8586
15	6.0772	5.8166

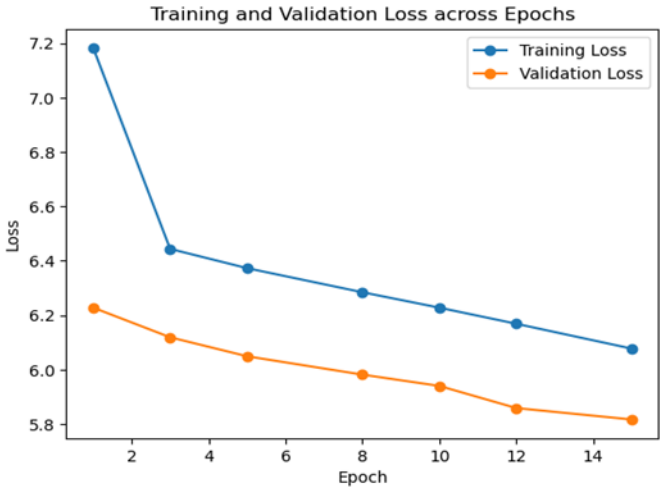

Figure 4. Training and validation loss curves across epochs illustrating the convergence behaviour of the proposed model.

The training and validation loss curves for various epochs are displayed in Figure 3. As training continues, a steady decline in both training and validation loss is shown, suggesting stable optimization and successful learning. Reasonable generalization and minimal overfitting are suggested by the comparatively small difference between the two curves. Prior to final evaluation using ROUGE metrics, these findings verify that the model converges correctly.

ROUGE-based Performance Evaluation

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics, which are common assessments for summarization assignments, are used to assess the quality of the suggested abstractive summarization approach. Exact-match measurements like accuracy are inappropriate since a particular input text may have more than one valid summary. As a result, only ROUGE-1, ROUGE-2, and ROUGE-L scores are used for model evaluation.

In order to assess content significance, ROUGE-1 calculates the unigram similarity between

the derived summaries and the source summaries.

ROUGE-2 captures phrase-level uniformity and local consistency by measuring bigram

overlap. ROUGE-L assesses sentence-level structure and fluency and is based on the longest common subsequence.

The suggested LSTM-based encoder–decoder model may provide coherent and meaningful abstractive summaries, as shown by the achieved ROUGE scores. Specifically, the ROUGE-1 and ROUGE-L scores show that the model successfully extracts important information while preserving structural coherence in the summaries that are produced. ROUGE-2 scores are somewhat lower, as would be predicted for abstractive summarization, and this is in line with results published in previous research.

The suggested method performs competitively for short-text, opinion-oriented abstractive summarization, according to the ROUGE-based evaluation. These outcomes support the efficacy of the training approach and model design covered in the earlier sections. Table 4 shows the ROUGE and BLUE scores of our model

Table 4 : ROUGE and BLEU Scores of the Proposed Model

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SUM	BLEU-4
Our Model (LSTM + Attn + Beam)	0.34	0.099	0.322	0.731	0.15

Training and validation loss values over epochs are recorded in Table 5 and shown in Figure 5 and 6 to examine the learning behaviour and reliability of the suggested model. As the amount of epochs grows, training loss and validation loss both steadily decline, demonstrating the model's successful optimization and convergence.

The model appears to have minimal overfitting and to generalize quite well to new data based on the comparatively minor difference between training and validation loss.

In order to show resolution performance during training, token-level training accuracy is only reported for diagnostic purposes. Nevertheless, accuracy is not regarded as a reliable assessment criterion for abstractive text summarization. Therefore, only the ROUGE-1, ROUGE-2, and ROUGE-L metrics—which are displayed in the table—are used to assess the final model performance.

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SUM	BLEU-4
ABS	0.280	0.087	0.246	0.613	0.12
ABS+	0.292	0.098	0.255	0.645	0.14
Our Model (LSTM + Attn + Beam)	0.34	0.099	0.322	0.731	0.15

Table 5: ROUGE & BLEU Comparison

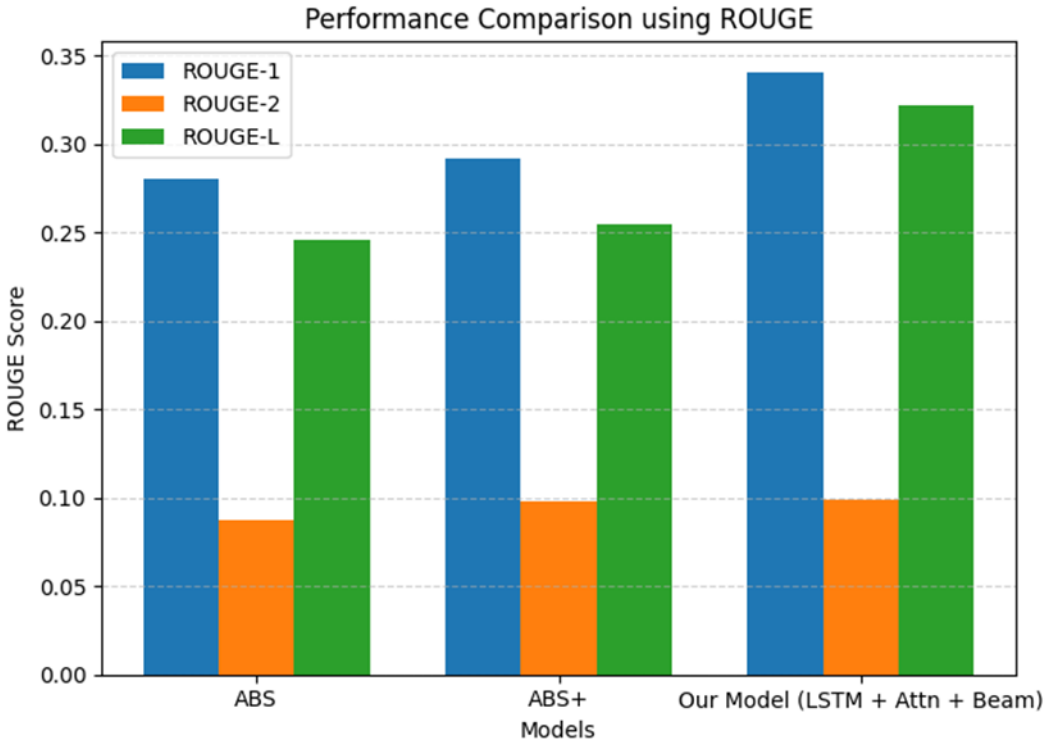

Figure 5:- Performance Compression using ROUGE

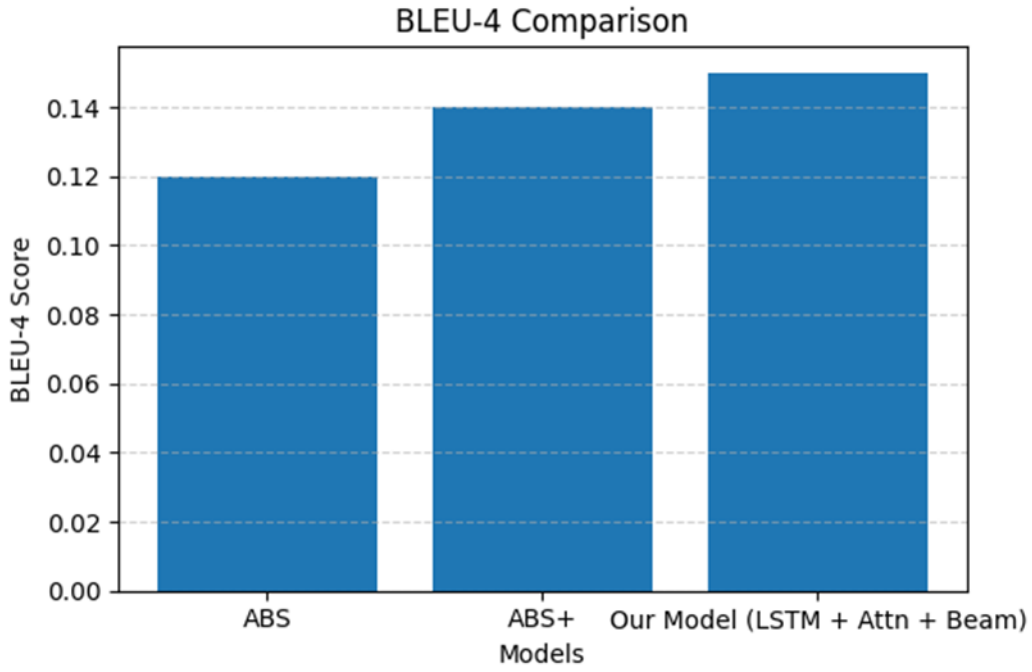

Figure 6 :- Performance Compression Using BLUE Score

Qualitative Analysis of Generated Summaries

Representative samples from the CNN/DailyMail dataset are shown below to offer meaningful analysis of the performance of the LSTM-powered encoder–decoder system using attention and beam search decoding. These instances demonstrate the model's capacity to extract important details and produce logical abstractive summaries. The examples are solely meant for qualitative examination.

Conclusion

An LSTM-powered sequence-to-sequence abstractive text summarization model with beam search decoding and an attention mechanism was reported in this study. By efficiently extracting contextual data from input text and creating summaries in an autoregressive fashion, the suggested architecture was created to produce compact and cohesive summaries.

ROUGE-1, ROUGE-2, and ROUGE-L metrics—all common and frequently used for summarization tasks—were used to assess the model. The results of the experiments show that the suggested method can produce clear and informative summaries, especially for situations involving brief texts and opinions. The model successfully retains salient material and reasonable structural coherence in the generated summaries, according to the ROUGE-based evaluation.

Apart from the quantitative assessment, training convergence study with training and validation loss curves demonstrated minimal overfitting and steady optimization performance. Additionally, qualitative examples demonstrated that, in contrast to greedy decoding, the combination of attention and beam search helps the model concentrate on crucial content and enhances summary coherence.

The state-of-the-art for large-scale news summarization is represented by contemporary Transformer-based models, however they frequently demand significant processing power. The suggested LSTM-based architecture, on the other hand, provides a computationally effective and lightweight substitute, which makes it appropriate for contexts with limited resources and domain-specific applications like review summarization.

Future research will concentrate on adding Transformer-based components, expanding the model with more sophisticated attention processes, and assessing performance on more extensive benchmarks. Investigating hybrid architectures and domain adaption strategies could enhance summary robustness and quality across various text domains.

References

[1] E. Hovy and C.-Y. Lin, “Automated text summarization in SUMMARIST,” Adv. Autom. text Summ., vol. 14, pp. 81–94, 1999.

[2] I. Mani, D. House, G. Klein, L. Hirschman, T. Firmin, and B. M. Sundheim, “The TIPSTER SUMMAC text summarization evaluation,” in Ninth Conference of the European Chapter of the Association for Computational Linguistics, 1999, pp. 77–85.

[3] S. K. Kurian and S. Mathew, “Survey of scientific document summarization methods,” Comput. Sci., vol. 21, 2020.

[4] H. P. Luhn, “The automatic creation of literature abstracts,” IBM J. Res. Dev., vol. 2, no. 2, pp. 159–165, 1958.

[5] H. P. Edmundson, “New methods in automatic extracting,” J. ACM, vol. 16, no. 2, pp. 264–285, 1969.

[6] R. Mihalcea and P. Tarau, “Textrank: Bringing order into text,” in Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 404–411.

[7] R. Nallapati, F. Zhai, and B. Zhou, “SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents,” in 31st AAAI Conference on Artificial Intelligence, AAAI 2017, 2017, pp. 3075–3081.

[8] Y. Liu and M. Lapata, “Text summarization with pretrained encoders,” arXiv Prepr. arXiv1908.08345, 2019.

[9] R. Barzilay and M. Lapata, “Modeling local coherence: An entity-based approach,” Comput. Linguist., vol. 34, no. 1, pp. 1–34, 2008.

[10] A. Nenkova and K. McKeown, “Automatic summarization,” Found. Trends® Inf. Retr., vol. 5, no. 2–3, pp. 103–233, 2011.

[11] A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for abstractive sentence summarization,” arXiv Prepr. arXiv1509.00685, 2015.

[12] S. Chopra, M. Auli, and A. M. Rush, “Abstractive sentence summarization with attentive recurrent neural networks,” in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 93–98.

[13] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gulçehre, and B. Xiang, “Abstractive text summarization using sequence-to-sequence RNNs and beyond,” CoNLL 2016 - 20th SIGNLL Conf. Comput. Nat. Lang. Learn. Proc., pp. 280–290, 2016, doi: 10.18653/v1/k16-1028.

[14] P. Gigioli, N. Sagar, A. Rao, and J. Voyles, “Domain-aware abstractive text summarization for medical documents,” in 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2018, pp. 2338–2343.

[15] M. Afsharizadeh, H. Ebrahimpour-Komleh, and A. Bagheri, “Automatic text summarization of covid-19 research articles using recurrent neural networks and coreference resolution,” Front. Biomed. Technol., 2020.

[16] S. N. Turky, A. S. A. Al-Jumaili, and R. K. Hasoun, “Abstractive Text Summary of COVID-19 Documents based on LSTM Method and Word Embedding.,” Webology, vol. 18, no. 2, 2021.
[17] F. Karim, S. Majumdar, H. Darabi, and S. Chen, “LSTM fully convolutional networks for time series classification,” IEEE access, vol. 6, pp. 1662–1669, 2017.
[18] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, “On faithfulness and factuality in abstractive summarization,” arXiv Prepr. arXiv2005.00661, 2020.
[19] Z. Abidin and A. Junaidi, “Text stemming and lemmatization of regional languages in Indonesia: a systematic literature review,” J. Inf. Syst. Eng. Bus. Intell., vol. 10, no. 2, pp. 217–231, 2024.
[20] C. P. Chai, “Comparison of text preprocessing methods,” Nat. Lang. Eng., vol. 29, no. 3, pp. 509–553, 2023.
[21] S. Das, S. B. Partha, and K. N. I. Hasan, “Sentence generation using lstm based deep learning,” in 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1070–1073.
[22] X. Rong, “word2vec parameter learning explained,” arXiv Prepr. arXiv1411.2738, 2014.
[23] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
[24] K. D. P. B. J. Adam, “A method for stochastic optimization,” arXiv Prepr. arXiv1412.6980, vol. 1412, no. 6, 2014.
[25] https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-1. python-keras/.