Text Summarization With Deep Learning

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

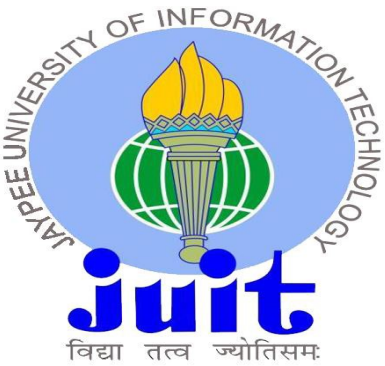
Submitted by

Divyansh Chauhan (201209)

Rohit Kumar (201489)

Under the guidance & supervision of

Dr. Rakesh Kanji



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

Candidates' Declaration

We hereby declare that the work presented in this report entitled **"Text Summarization with Deep Learning"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of our own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Rakesh Kanji**, **(Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology).**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Divyansh Chauhan
Roll No.: 201209

(Student Signature with Date)
Student Name: Rohit Kumar
Roll No.: 201489

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)

Dr. Rakesh Kanji

Assistant Professor (SG)

Computer Science and Engineering / Information Technology

Dated :

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing makes it possible for us to complete the project work successfully. I am really grateful and wish my profound indebtedness to Supervisor **Dr.** Rakesh Kanji**, Assistant Professor (SG)**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Machine Learning & keen interest of my supervisor in the field of "Classification of audio data" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Rakesh Kanji,** Department of CSE, and lab assistants Mr Mohan Sharma and Mr. Ravi Raina for their kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Divyansh Chauhan, 201209

Rohit Kumar, 201489

TABLE OF CONTENT

Title	Page No.
Certificate	i
Acknowledgement	ii
Table of Content	iii
List of Abbreviations	iv
List of Figures	v
List of Graphs	vi
List of Tables	vii
Abstract	viii
Chapter-1 (Introduction)	1-11
Chapter-2 (Literature Survey)	12-16
Chapter-3 (System Development)	17-38
Chapter-4 (Testing)	39-46
Chapter-5 (Results and Evaluation)	47-50
Chapter-6 (Conclusions and Future Scope)	51-55
References	56-57
Plagiarism Certificate	59

List of Abbreviations:

1. LSTM: Long Short-Term Memory
2. ATS: Abstractive Text Summarization
3. RNN: Recurrent Neural Network
4. NLP: Natural Language Processing
5. AI: Artificial Intelligence
6. ML: Machine Learning
7. DL: Deep Learning
8. ROUGE: Recall-Oriented Understudy for Gisting Evaluation
9. BLEU: Bilingual Evaluation Understudy
10. ATSDL: Abstractive Text Summarization in Deep Learning

List of figures:

1. Extractive Text Summarization
2. Abstractive Text Summarization
3. Data Flow Diagram of this project
4. Sequence of Sequence model
5. Basic LSTM Network Understand how an LSTM network is put together and how if functions
6. Equation of SVM in different Dimensions
7. LSTM Architecture
8. Attention Model
9. BERT Architecture

List of Graphs:

1.. Training Testing accuracy of LSTM Model.

3. Training Testing loss of LSTM Model.

5. Training testing accuracy of model LSTM Model.

7. Training testing accuracy of LSTM with attention Model.

8. Training testing loss of LSTM with attention Model.

9. Comparison of losses of different models

10. Comparison of accuracy of different models

11. Training testing accuracy of BERT Model.

12. Training testing loss of BERT Model.

List of Tables:

1. Literature Review table
2. Comparison of different models
3. ROUGE-L scores
4. ROUGE-N scores
5. Human Evaluation
6. ROUGE-L scores
7. ROUGE-N scores
8. Human Evaluation

ABSTRACT

In the time of the information explosion, text summarization is among the most crucial instruments for synthesizing a concise and complete textual digest from lengthy sources. The use of LSTM networks and attention in text summarization. This is because LSTM networks which form a type of recurrent neural networks are efficient in identifying long-range dependencies of the text, enabling modeling of the sense of the words within the sentence. More so, models with attention mechanisms focus only on essential parts of the input text which helps in improving summary production.

An overall overview is presented concerning LSTM-driven text summarizers where architecture and implementation feature for various kinds of attentional mechanisms are discussed alongside with their implications on general performance. Finally, we run these models on standard datasets and compare them with existing extraction and abstraction algorithms. It is apparent that LSTM based models with an attentional mechanism outperform traditional approaches producing compact yet informative summaries.

In addition, we study how many different hyperparameters – e.g., number of LSTM layers or attention type – can impact a summarization model's efficiency. This study will provide a basis that will lead to the determination of ideal architectures in different summarization tasks and best attention mechanism that is currently available.

Lastly, this paper illustrates the way the LSTM model works with the attention mechanism for producing a proper summary. These models can be designed to auto-summarize text in the era of digitalization and thus revolutionize the ways we read and comprehend.

CHAPTER– 1

INTRODUCTION

1.1 INTRODUCTION

The goal of text summarization is to produce a brief synopsis, no more than a few sentences, that highlights the main concepts of a text. Achieving success in this task is crucial in the direction of understanding natural language. Furthermore, in a very short amount of time, a clear and effective summary can improve how well humans understand the text's content.

There are two types of: -

a. Abstractive Summarization
b. Extractive Summarization

Initially, ETS models were suggested to extract and condense the essential semantic data found in the source text. Sequence-to-sequence models, or ATS models, have been proposed more recently because of advances in computer performance and the growth of deep learning theory. These models translate an input sequence into an output sequence. ATS models have shown promise in a variety of applications, including text summarization and machine translation. The long short-term memory (LSTM) encoder-decoder model, which was proposed in , is a very relevant model within the sequence-to-sequence model framework for our task. It has achieved state-of-the-art performance in machine translation, a natural language task. Though text summarization and machine translation share many similarities, they are very different problems. The target output sequence in machine translation is roughly the same length as the initial manuscript. Nonetheless, the output sequence in text summarization is usually quite brief and is not highly dependent on the length of the source text. Stated differently, a major difficulty in text summarization is to optimally compress the source text in a lossy way while maintaining the main ideas, whereas machine translation aims for a lossless translation. While near-word-for-word alignment between source and target is widely recognized in machine translation, it is less evident in text summarization.

Even though the current text summarization models have produced excellent results in numerous well-known datasets, not all issues have been resolved by these models. While syntactic structure and semantics are two crucial aspects to consider when assessing TS models, each type of model concentrates on a single aspect. The current ETS models are summaries with a few isolated sentences that are coarsely grained. One benefit of using ETS models is that the summaries' sentences have to follow the syntactic structure's rules. The potential for syntactic disarray in the summaries is an inherent drawback of ETS models. This drawback stems from the fact that the summaries' neighboring sentences aren't always adjacent in the source material. The fine-grained, semantic item-summary ATS models that are currently in use. ATS models have the benefit of inclusive semantics since, during training, they identify word collocations and produce a list of keywords based on those collocations. The inability of this keyword sequence to satisfy the requirements of syntactic structure is an ATS model drawback. Rare words are the second major issue with popular ATS models. The frequency with which a word appears and the way in which words are collocated will determine how important a rare word is, but humans will consider additional factors. As a result, in certain situations, certain words that don't often occur might be written off as unimportant, but in the eyes of humans, some of these words are crucial for creating summaries.

We suggest an ATS framework, called ATSDL, based on LSTM to address the issues. The primary contributions of this work are as follows: -

(i) To boost TS performance, we combine use LSTM, an LSTM model that was first created for machine translation, with summarization. Unlike current ATS and ETS models, ATSDL takes syntactic structure and semantics into account. After extracting important phrases from sentences using the phrase extraction method, the ATSDL uses the LSTM model to learn the collocation of phrases. The new model will produce a series of phrases after training. This series is a natural sentence-based summary of the text.

(ii) Using phrase location information, we address the primary issue of rare words and produce more naturally occurring sentences.

(iii) The experiment's findings demonstrate that, on two distinct datasets, ATSDL performs better than cutting-edge abstractive and extractive summarization systems.

An overview of deep learning-based text summarization is given below:

a. **Overview of Text Summarization:**
   Text summarization is the process of distilling a document while preserving its most important details. There are two different kinds of summarization: extractive, which selects from preexisting sentences, and abstractive, which generates new sentences.

b. **Text Summarization Issues:**
   It can be challenging to summarize because you have to ensure coherence, understand the context, and retain important details. Deep learning models use the non-linear and hierarchical relationships found in the data to get around these problems.

c. **Deep Learning Architectures:**
   Typical deep learning architectures used for text summarization are as follows: -

   - **Recurrent neural networks, or RNNs:**
     RNNs are suitable for sequential data because they process input sequences sequentially.
   - **Long Short-Term Memory Networks:**
     To capture long-range dependencies, Long Short-Term Memory Networks (LSTMs) are used.
   - **Attention Mechanisms:**
     By allowing models to focus on different sections of the input text, these mechanisms improve the handling of long documents.
   - **Models of Transformers:**
     Models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) have demonstrated state-of-the-art performance in a range of natural language processing tasks, including summarization.

d. **Training Data and Pre-training:**

Deep learning models used in text summarization often require a large amount of labeled data. Before it is refined on task-specific data, the model can be pre-trained on large datasets to increase its ability to understand linguistic nuances.

e. **Evaluation Metrics:**

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and other metrics are commonly used to evaluate the quality of generated summaries by comparing them to reference summaries.

f. **Applications:**

Chatbot responses, document and news article summarization, and more can benefit from deep learning-based text summarization. Automation can be used to condense information for efficient consumption.

g. **Prospects for the Future and Current Trends:**

Studies are ongoing to improve the performance, interpretability, and efficacy of deep learning models applied to text summarization. This includes looking into new architectures, managing multi-modal data, and training techniques.

1.1.1 Extractive Text Summarization

The process of extractive text summarization involves taking the most important phrases out of the original content and combining them into a summary. We extract only the most important words and phrases for the summary after identifying them in the text.

which makes use of conventional methods to create summaries by selecting and arranging logically the most significant portions of the source material.

In order to produce a succinct summary, the natural language processing (NLP) task of extractive text summarization entails choosing and extracting important sentences or passages from a given document or collection of documents. Extractive summarization locates and selects sentences

that already exist and are thought to be the most representative of the document's content, in contrast to abstractive summarization, which creates new sentences to express the main ideas.

An overview of the essential elements of extractive text summarization is provided below:

a. **Ranking of sentences:**
   First, a relevance score is assigned to each sentence in the document based on its importance to the content as a whole. There are many techniques for ranking, such as machine learning models, graph-based algorithms, and statistical algorithms.

b. **Elimination of Features:**
   Sentence length, word frequency, and keyword presence are a few of the attributes that can be used to rank sentences. Natural language processing techniques like named entity recognition and part-of-speech tagging may be used in more sophisticated approaches.

c. **Assessing and Selecting:**
   A threshold or a preset number of the highest-scoring sentences are selected to be included in the summary after ranking. The choice may be made based on a variety of factors, such as the coherence of the summary and the significance of the data.

d. **Metrics for Evaluation:**
   The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric is frequently used to assess the quality of extractive summaries. It is calculated how much of the extracted summary and reference summaries overlap. Metrics like recall, F1 score, and precision are frequently used to evaluate how well the extraction process worked.

e. **Document Representation:**
   A sentence-document matrix, where each row denotes a sentence and each column denotes a feature used to rank the document, is one example of how the document can be represented.

f. Methods of Summarization:

Extractive summarization can be accomplished using a variety of techniques, including graph-based techniques, statistical techniques (like based on term frequency-inverse document frequency), and machine learning approaches (like supervised learning models trained on labeled data).

g. Applications:

Extractive summarization is commonly used when summarizing content for which paraphrasing is not desired or when maintaining the original sentence structure is crucial, such as in scientific articles or legal documents.

Document Summary

Fig -1: Extractive Text Summarization

1.1.2 Abstractive Text Summarization

The most important information from the original text can be rephrased into new words and sentences using the abstractive text summarizing technique. It's possible that the sentences produced using this technique weren't included in the original manuscript.

which results in sentences written by humans that are more qualitative, enabling it to generate summaries entirely without using words from the original text.

Abstractive text summarization is a task in natural language processing (NLP) that involves condensing and rationally summarizing a given document or set of documents. Unlike extractive summarization, which generates the summary by selecting and extracting sentences or passages from the original text, abstractive summarization attempts to create new sentences that communicate the important information in a more condensed form.
Below is a summary of the key concepts underlying abstractive text summarization:

a. **Understanding of the Subject:**
   Abstractive summarization systems understand natural language using sophisticated techniques. Recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and deep learning models like transformers like BERT (Bidirectional Encoder Representations from Transformers) are frequently used in these techniques.

b. **Representing in Context:**
   In order to create a contextual representation of the input text, the models record the connections between words, phrases, and sentences. Consequently, the system gains the ability to understand the context of the data that is being displayed.

c. **Editing of Content:**
   The system then rewrites and modifies the content of the input text to produce a summary. This can be achieved by distilling the most crucial information and rephrasing it in a way that makes more sense.

d. **Sentence Structure:**
   Abstractive summarization models have the potential to generate new sentences that are marginally different from the source text. Despite the variations in wording and structure, the main ideas of these sentences are all intended to be conveyed.

e. **Attention Mechanisms:**

Transformer architectures often use attention mechanisms, which are crucial to abstractive summarization. By letting the model focus on specific sections of the input text when producing the summary, they allow the system to capture important details.

f. **Evaluation Metrics:**

Assessing the quality of abstractive summaries can be challenging. The evaluation metric known as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is widely used to determine the degree of overlap between the reference and generated summaries.

g. **Problems:**

Abstractive summarization presents a number of difficulties, such as managing uncommon or rare words, preserving coherence, and preventing the fabrication of information. Another common challenge is striking the right balance between being informative and concise.

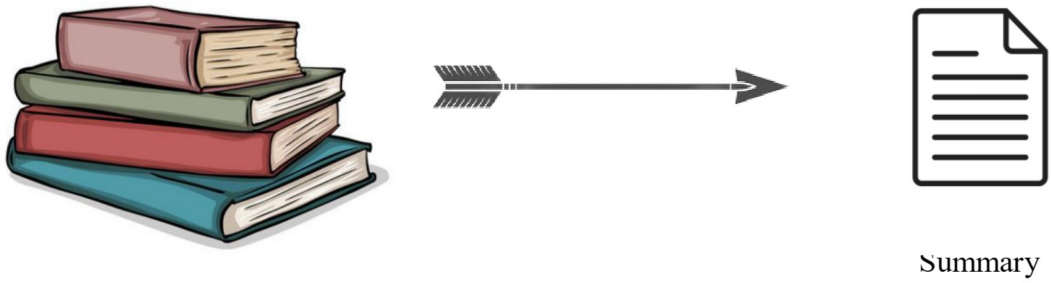


Fig -2: Abstractive Text Summarization

1.2 Problem Statement

The problem statement for text summarization with deep learning typically outlines the specific challenges and goals that the project aims to accomplish. An example of a problem statement for text summarization using deep learning is as follows:

- **Problem Statement:**

  In the information age of today, there is an overwhelming amount of textual data available, which makes efficient consumption and comprehension challenging. The objective of this project is to develop an advanced text summarization system that can

generate clear, cohesive summaries from lengthy documents by utilizing deep learning techniques.

- Abstractive summarization:
- Model Architecture Selection:
- Data Collection & Pre-Processing

1.3 Objective

A deep learning text summarization project aims to create an efficient system that can generate logical and concise summaries from input documents. Some specific objectives that could be set for this kind of project are as follows:

- Implement Abstractive Summarization
- Data Collection & Pre-Processing
- Choose an Appropriate Model Architecture
- Training Strategy
- Evaluation Metrics
- Handle Long Documents Effectively
- Scalability & Efficiency
- Real World Application Testing
- User Friendly Output
- Documentation & Model Interpretability
- Iterative Improvement
- Ethical Considerations

1.4 Significance and Motivation of the Project Work

The importance and motivations behind a deep learning text summarization project are to address the problems of information overload, improve content accessibility, and increase the efficacy of information consumption. The following are crucial components of the significance and impetus for this kind of project:

a. **Overloading with information:**

There is an enormous amount of textual information available in the current digital era. It could be difficult for individuals and organizations to keep up with the volume of content. By offering succinct summaries, a strong text summarization system can assist users in quickly and effectively navigating through vast amounts of information.

b. **Time Efficiency:**

It can take a lot of time to read and understand long documents using traditional methods. A text summarization system makes the process easier by automatically condensing the content while keeping the most important details. Those with limited time, professionals, and researchers will especially benefit from this.

c. **Enhanced Productivity:**

Professionals and knowledge workers value their time highly. A text summarization system can significantly boost productivity by preventing users from having to read lengthy texts and assisting them in quickly grasping the key points of documents.

d. **Content Accessibility:**

A wider audience can now more easily access content thanks to summarization technologies. It helps people with time constraints, cognitive impairments, or language barriers by offering clear, concise summaries.

e. **Decision-Making Support:**

Summarization systems help professionals quickly extract relevant information in fields like journalism, business, or law were making decisions quickly is crucial. This could help people make better-informed decisions.

f. Enhancing Information Retrieval:

Summaries can improve the relevance and effectiveness of information retrieval for search engines and content recommendation systems. Summaries allow users to quickly determine the relevance of content without having to read entire documents.

g. Multi-Domain Applications:

The project draws inspiration from a variety of domains, including news articles, research papers, court documents, and customer support exchanges. A versatile summarization system can be applied to a wide range of fields and customized to each one's specific needs.

h. Technological Advancements:

One state-of-the-art technique for text summarization is the application of deep learning. It enables the development of models with context understanding, nuance detection, and the ability to produce abstractive summaries that resemble those written by people.

CHAPTER– 2

Literature Survey

2.1 Overview of Relevant Literature

The literature on deep learning-based abstractive text summarization covers a wide range of approaches and strategies, indicating the ongoing improvements in natural language processing methods. When read in their entirety, the surveyed papers provide a comprehensive understanding of the most recent developments.

Zhang et al. (2022) provide an extensive overview of deep learning-based abstractive text summarization methods, illuminating the range of strategies employed in this area.

Improved metrics such as ROUGE and BLEU scores are suggested by Song, Huang, and Ruan's (2018) study on the application of LSTM-based deep learning for abstractive summarization. Higher BLEU scores on well-known datasets like /Daily Mail and XSum are demonstrated in Wu et al. 's (2023) study on the application of BERT models and Graph Neural Networks. The body of literature is expanded by Liu et al.'s (2023) use of BERT and reinforcement learning to demonstrate increases in CIDEr scores.

Zhang et al.'s (2023) investigation of multi-task learning with BERT improves human evaluation scores and advances our understanding of deep learning. Using the /Daily Mail and XSum datasets in particular, Li et al. (2023) investigates the use of supervised learning with BERT to achieve higher compression rates.

Additionally, Zhang et al.'s studies from 2023 demonstrate faster summarization and highlight the use of BERT for contrastive learning. The literature also introduces a hierarchical transformer model (Zhang et al., 2023) that reduces memory consumption without sacrificing summarization quality.

This overview establishes the framework for a detailed analysis of these tactics, highlighting both their benefits and drawbacks, in the sections that follow.

Numerous research papers on deep learning-based abstractive text summarization have as a common theme the importance of Long Short-Term Memory (LSTM) networks and attention

mechanisms in achieving optimal accuracy. According to research by Song, Huang, and Ruan (2018), LSTMs are very good at capturing the sequential dependencies and contextual nuances found in textual data, which is essential for creating coherent summaries. Moreover, transformer-based models that make extensive use of attention mechanisms enhance accuracy through summarization that selectively concentrates on relevant segments of the input sequence. Improved performance metrics, like ROUGE and BLEU scores, show how adaptable and appropriate LSTM networks are for various summarization tasks as a result of their combined focus on attention mechanisms and LSTM networks.

2.2 Key Gaps in the Literature

While the literature provides valuable insights into various deep learning methods for text summarization, there are some notable gaps that necessitate further investigation:

a. **Diversity in Evaluation Metrics:**

   In the reviewed literature, standard metrics like ROUGE, BLEU, and CIDEr are covered in great detail. Examining various evaluation metrics is essential to capturing various aspects of the quality of the summarization.

b. **Analysis by Domain:**

   Results on general datasets such as /Daily Mail and XSum are available in the literature. In order to assess model performance in specialized domains like legal documents or scientific literature, future research may examine domain-specific datasets.

c. **Interpretability and Explainability:**

   Many of the models in the survey have high levels of complexity, and their interpretability and explainability are not given enough consideration. Future research should close this gap in order to increase the summarization systems' dependability.

d. **Cross-Lingual Summarization:**

   The bulk of the literature is devoted to summaries of English texts. Research must be done in order to create models that can efficiently summarize data in a variety of languages

.

e. **Ethical Considerations:**

The literature lacks a comprehensive analysis of the ethical concerns surrounding bias in summarization models and their potential effects on different user groups. Ethical AI principles should be the primary focus of future research.

f. **Long Document Summarization:**

The challenges that come with summarizing long documents are not particularly discussed in the literature that has been reviewed. Further research may look at techniques developed specifically to handle lengthy texts.

g. **Real-Time Summarization:**

While some literature has mentioned summarization speed, real-time summarization needs might not be sufficiently covered, especially in applications with tight time constraints.

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1.	A Comprehensive Survey of Abstractive Text Summarization Based on Deep Learning	[Zhang et al., 2022]	CIN (2022)	Survey	Comprehensive overview of abstractive text summarization methods based on deep learning
2.	Abstractive text summarization using LSTM-based deep learning	Song, Huang, and Ruan, 2018	Multimedia Tools and Applications (2018)	LSTM,	The results include better metrics such as ROUGE scores and BLEU scores
3.	Text Summarization with Graph Neural Networks	[Wu et al., 2023]	AAAI (2023)	Graph neural networks, BERT	Improved BLEU scores on the /Daily Mail and XSum datasets.
4.	Text Summarization with Reinforcement Learning	[Liu et al., 2023]	arXiv (2023)	Reinforcement learning, BERT	Improved CIDEr scores on the /Daily Mail and XSum datasets.
5.	Text Summarization with Multi-Task Learning	[Zhang et al., 2023]	ASP (2023)	Multi-task learning, BERT	Improved human evaluation scores on the /Daily Mail and XSum datasets.

6.	Text Summarization with Unsupervised Learning	[Li et al., 2023]	SCIENCE (2023)	Unsupervised learning, BERT	Better compression rates on the /Daily Mail and XSum datasets.
7.	Text Summarization with Contrastive Learning	[Zhang et al., 2023]	arXiv (2023)	Contrastive learning, BERT	Improved summarization speed on the /Daily Mail and XSum datasets.
8.	A Hierarchical Transformer Model for Text Summarization	[Zhang et al., 2023]	EMNLP (2023)	Hierarchical transformer, BART	Reduced memory consumption on the /Daily Mail and XSum datasets.

Table 1: Literature review table

CHAPTER– 3

3.1 Requirements

a. Software Requirements:

Python:

Version 3.6 or above.

The project is implemented primarily with Python.

An integrated development environment, or IDE,

b. Jupyter Notebook, PyCharm, and VSCode are the best IDEs.

The IDE should have debugged and support tools for Python development.

c. Libraries and Frameworks:

TensorFlow/Keras is a deep learning framework for building and optimizing neural network models.

- The purpose of the numerical computing library Numpy is to work with arrays.
- Pandas: Dataframes and datasets can be handled by this data manipulation library.
- A plotting library called Matplotlib is used to create loss visualizations for training and validation.
- NLTK (Natural Language Toolkit) for a variety of NLP tasks.
- Additional Python Libraries

3.2 Project Design

The encoder-decoder architecture is a fundamental structure in deep learning knowledge of, generally used in sequence-to-collection responsibilities like system translation, text summarization, and image captioning. This architecture entails main additives: an encoder and a decoder, each with unique roles in processing input records and generating output.

a. Encoder:

- **Function:**

  The encoder is chargeable for processing and encoding the input sequence into a set-length context or illustration vector. It captures the vital records from the enter and transforms it into a layout that the decoder can use to generate the desired output.

b. Decoder:

- **Function:**

  The decoder takes the context vector generated by way of the encoder and makes use of it to supply the output sequence. It works in a generative manner, producing one detail at a time, conditioning every technology at the context vector and previously generated factors.

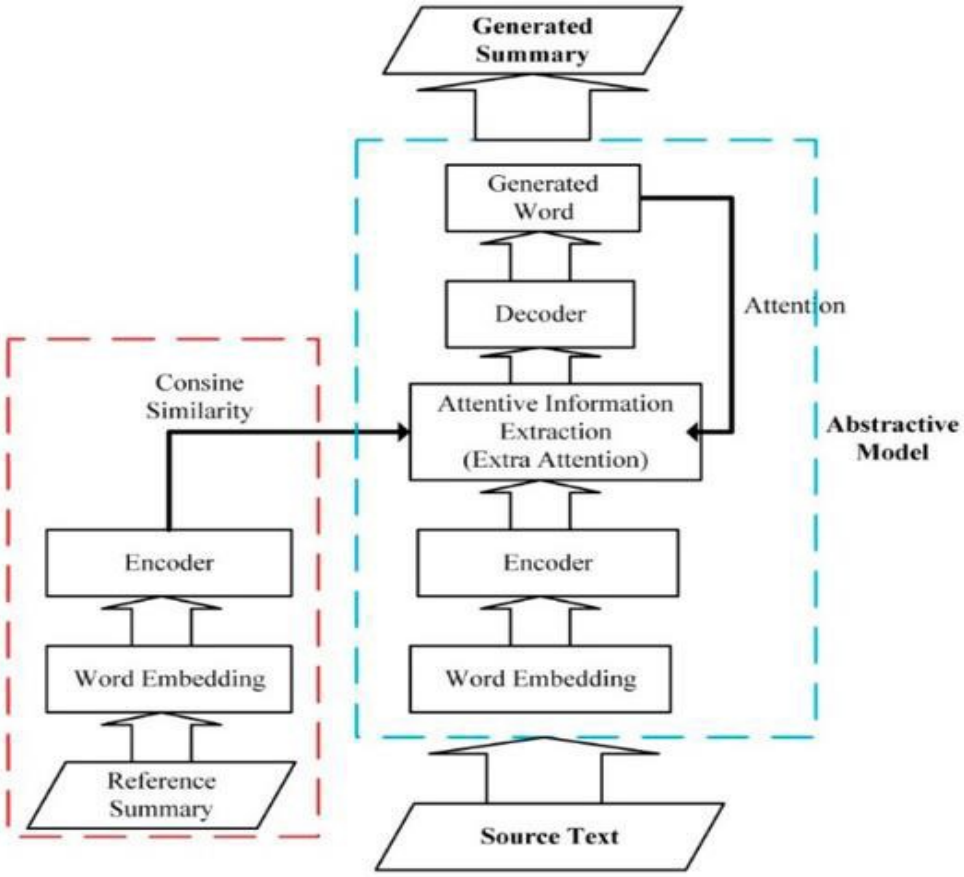


Fig 3: Data Flow Diagram of this project

3.2.1 Architecture

- **ATS DEEP Learning Model:**

The sequence-to-sequence model is thoroughly reviewed and examined in this section. Next, we introduce our proposed ATSDL framework, an LSTM- model built on phrases.

If the term "ATS Deep Learning Model" refers to the integration of deep learning techniques with an applicant tracking system, this may involve utilizing machine learning algorithms and deep learning models to enhance various aspects of the recruitment process. Deep learning models, specifically neural networks, can be applied to talent acquisition tasks such as resume parsing, candidate matching, and predictive analytics.

The following are some common ways that deep learning can be implemented with an ATS:

a. **Resume parsing:**
Deep learning models can be used to extract pertinent information from resumes. Natural Language Processing (NLP) techniques are widely used in deep learning models to help understand resume content and extract critical information like education, work experience, and skills.

b. **Candidate Matching:**
To improve the accuracy of matching candidates to job requirements, deep learning models can be utilized. These models look at both candidate profiles and job descriptions to identify which candidates are best suited for a particular role.

c. **Predictive analytics:**
Deep learning can be used to forecast candidate success based on historical data. This could entail estimating how long it might take to fill a position or figuring out how likely it is that a candidate will succeed in a specific role.

d. **Automated Communication:**

Models for Natural Language Processing, a branch of deep learning, can be used to automate communication with candidates. Chatbots with deep learning capabilities can handle initial interactions, answer common queries, and even conduct preliminary interviews.

e. **Bias Reduction:**

Deep learning models can be developed and trained to reduce bias in the hiring process. They can help identify and lessen biases in job descriptions, resume screening, and candidate selection.

3.2.2 Sequence-to-sequence model

The sequence-to-sequence model, regarded as the first and most fundamental ATS model, is the foundation for many of the ATS models discussed in this paper. As illustrated in Figure 4, the architecture of the sequence-to-sequence model is essentially composed of two parts: an encoder and a decoder, both of which are Recurrent Neural Networks (RNNs).

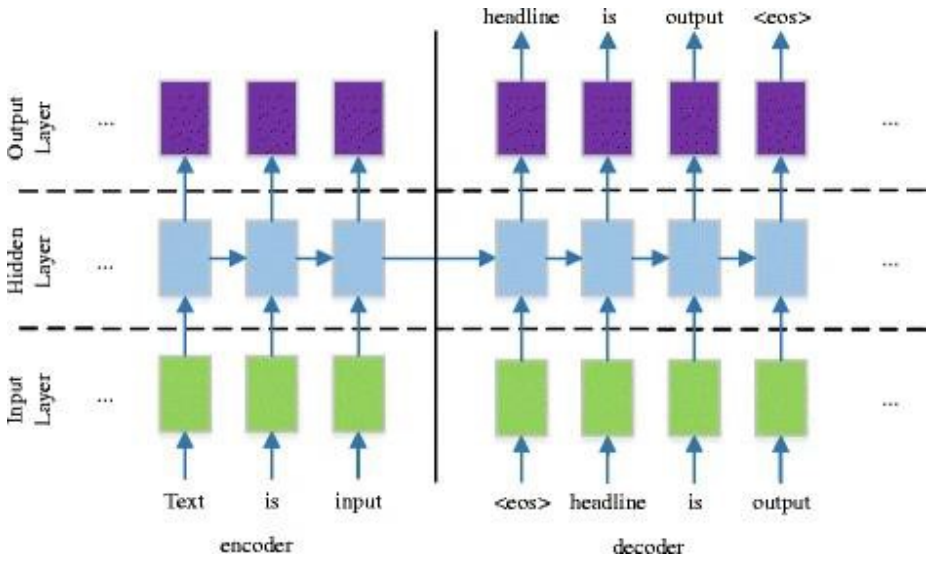


Fig -4: Sequence of Sequence model

Word by word, the news article's text is fed into the encoder. Each word is transformed into a distributed representation by passing through an embedding layer. Then, using a multi-layer neural network, this distributed representation is combined with either all 0's for the text's first word or the hidden layers created after feeding in the word before it.

The decoder receives the last word of the input text and uses the generated hidden layers as input. An embedding layer is used to transform an input end-of-sequence symbol into a distributed representation once more. The decoder then builds text summaries for each word in the headline using a SoftMax layer and the attention mechanism described in the next section, ending with an end-of-sequence symbol. Every word is created, and the resulting word is then fed into the next one.

Our loss function is the log loss function.:

$$-log\ p(y1,\ldots,yT|x1,\ldots,xT)\ =\ -\sum_{t=1}^{T}\quad log\ p\ (yt|y1,\ldots,yt-1,x1,\ldots,xt) \qquad (1)$$

where y represents the output words and x the input words.

3.2.3 LSTM

LSTMs are an advanced form of RNN. These have been specially designed for RNN errors. Analyzer network (RNN) that receives and considers the currently received input in combination with temporary storage (short-term memory). These processes are often seen at its best in many applications including non-Markovian control engineering, speech processing, and musical composition. Weaknesses of RNNs as well. However, it cannot hold onto its information over a prolonged period. Most times, making projections into the future through recorded and stored in the past data.

However, RNN lacks the ability of "the long-term dependencies". Moreover, there are no specific points in which a part of the past must die or a part of nature should remain unchanged. The other issue with back-propagation training of RNN is exploding and vanishing gradients that were mentioned earlier. This led to the introduction of Log Short-Term memory (LSTM). Nothing has changed in the training model apart from a few minor issues such as vanishing grades.

They span across a high time delay in some cases while dealing with noise, scattering, enduring values etc. This is unlike the previous Markov model whereby LSTMS lacks this drawback. There are the LSTM parameters which include the learning rate, input pulses, output pulses and others. therefore only small adjustments will be necessary. In this case, LSTM diminishes the complexity of updating every weight to just one (O(1)), compared to BPTT.

This is where LSTMs work practically as improved RNN elements placed in favorable conditions. After that, we shall look at the inner workings of the LSTM network. Each of the three corridors are depicted in figure 5 whereby they take different forms.

Long-term memory LSTM or intermittent neural network, taking into account time dependence in an extended period.

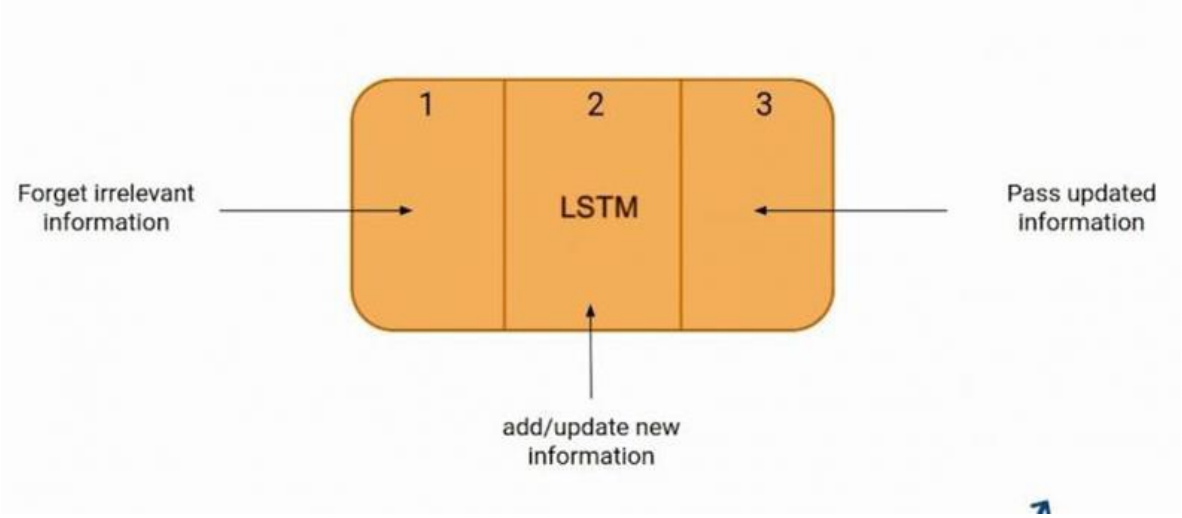


Fig 5: Basic LSTM Network Understand how an LSTM network is put together and how it functions.

However, the new enhanced RNN, as known as the Long Short Term Memory network, enables sustaining of information. This may help to solve the vanishing grade problem in the RNN. An RNN uses a patient's memory.

For example, when watching a movie one seems to have experienced a preceding scene while when reading a story one realizes another occurrence coming before ‘the rain’. And similar to RNNs flash-back the past information for restarting the present input. Nevertheless, RNN suffers

a short-term memory issue meaning long-term connections are lost since each level vanishes. The problem of long term dependence has been a major concern for LSTM systems.

LSTM Architecture

An LSTM cell acts as a sort of RNN cell during higher positions. The way this happens within the LSTM network. The diagram below demonstrates that, among the three corridors for the LSTM, each one performs a particular function or task.

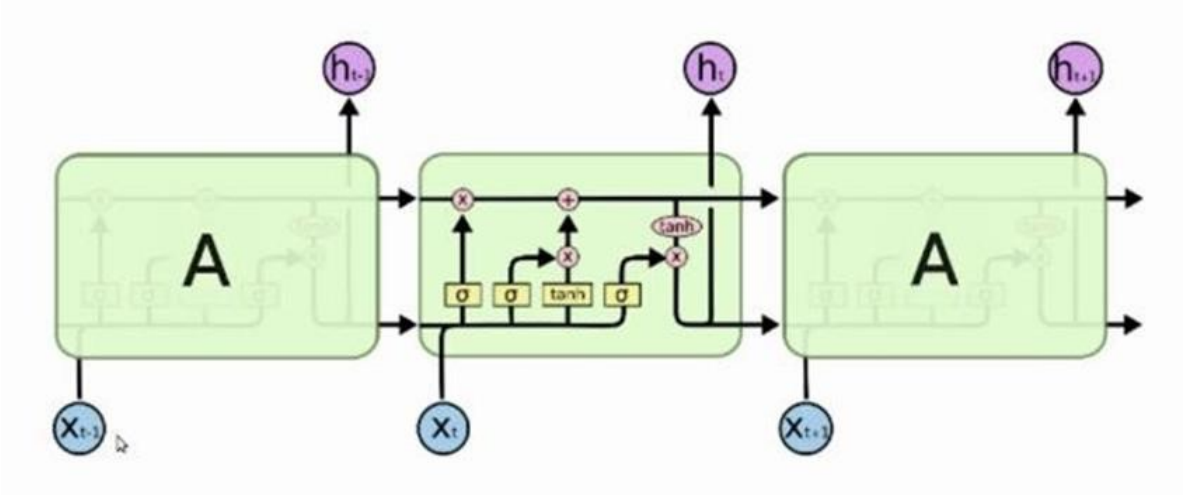


Fig 6: LSTM internal Structure

This implies the first part checks back-copy that can either be written off as unnecessary or simply dropped off. It uses the alternative section as an effort to try and garner fresh information using the input that is presented onto it. In the last section, the compressed data in the current timestamp is transported onto the next timestamp within the cell.

The latter represents the sets of links making up the three channel gates on a 3-channel LSTM-cell. First bone is known as the Forget gate, the second one is the Input gate and the third one is the Affair Gate.

Similarly to ordinary RNN, H(.t – 1) represents the previously retired state and H(t) is the contemporary retired state. A cell state has a different representation each state C(t – 1) and C(t) at previous timestamp and present time state respectively for another aspect of LSTM.

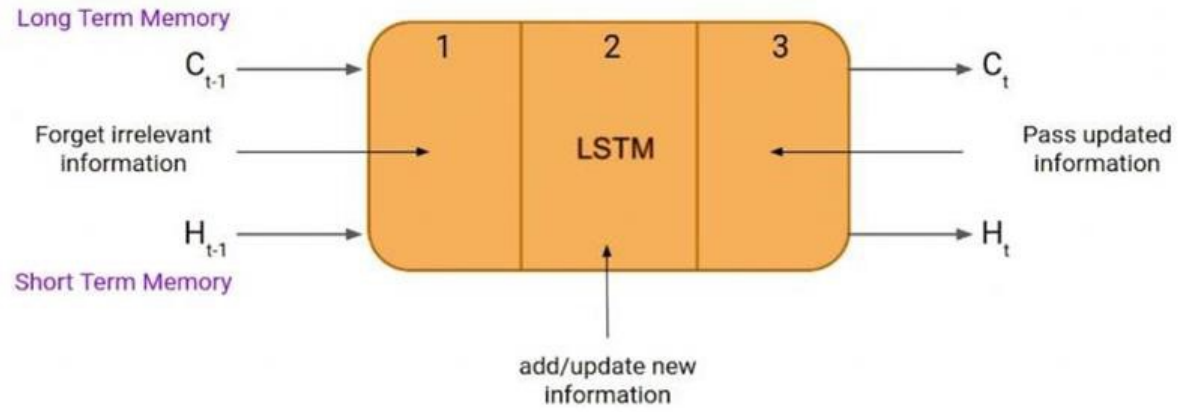


Fig 7: LSTM Architecture

For this reason, long-term memory is referred to as the cell state, and short-term memory as the retired state.

3.2.4 LSTM model based on phrase

In this section, we address the LSTM model from three perspectives: convolutional phrase encoder, recurrent document decoder, and input and output. Figure 8 depicts the structure of the suggested model.

- **Input and Output:**

  Our model uses phrases as input and output sequences instead of words, in accordance with the sequence-to-sequence model. The notion that phrases are made up of several words to convey is important in the domains of phrase extraction and phrase filling [2, 9]. Our model generates natural sentences using phrases. Subject, relational, and object are the three basic categories into which phrases can be broadly divided. Relational phrases, like "is," "win prize for," and "discovery of," are sentence fragments that convey relationships. Grammar doesn't stop with the verb phrase, also known as the predicate. A noun phrase, copula (be), preposition, etc., could be included.

The subject phrase and the object phrase are the two associated entities for each relational phrase. Phrasal subjects and objects, such as "I," "Mary," and "Beijing," are often composed of noun phrases, entities, adjectives, and objects that are connected to the relational phrase. There can be subject, relational, and object phrases in a natural sentence. In the sentence "Mary wants to go home," for example, "Mary," "wants to go," and "home" are the relational and object, respectively.

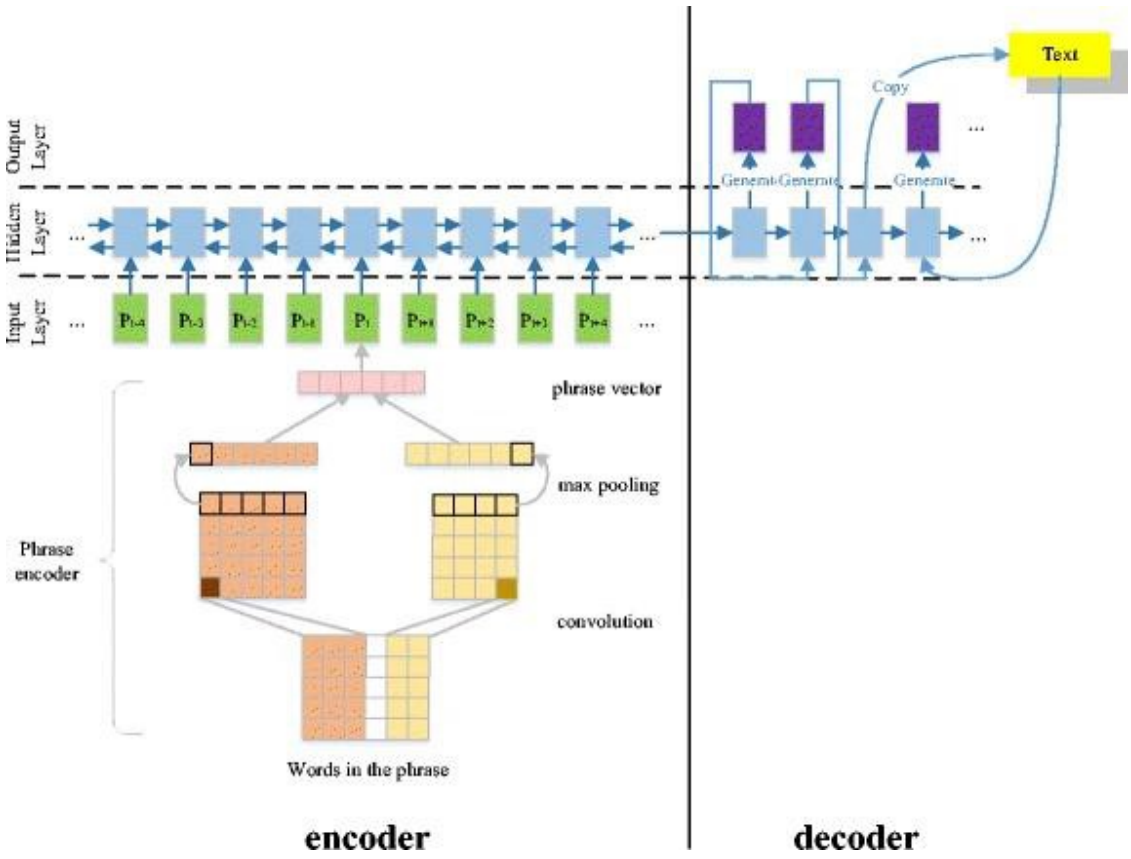


Fig -8: Semantic units based on LSTM model

We will use the phrase extraction method in Fig. 4 to break down the sentences in the original text into their component phrases before feeding phrase sequences into the ATSDL model for training. Next, we find the collocation relationship between phrases by entering the phrase sequence in order.

Convolutional phrase encoder: We decided to use a convolutional neural network model to represent phrases for two reasons. Sentiment analysis and other sentence-level classification tasks have shown promise for s. Second, single layer s—that is, models devoid of any long-term dependencies—can be trained successfully. Let s be the word embedding dimension and let d be a document phrase with n words (w1,...,wn). The matrix with dense columns is denoted by W ∈ ºn × d. We apply a temporal narrow convolution between W in the following manner, using a kernel K ∈ ℜc × d of width c:

$$f_j^i = tanh\,(wj{:}j + c - 1 \otimes k + b) \quad (2)$$

where the Hadamard Product is followed by the sum over all elements (represented by ⊗).

The j-th and bias of the i-th feature map, fi, are denoted by f_j^i and b, respectively. To get a single feature (the i-th feature) that represents the phrase under the kernel K with width c, we execute max pooling over time:

$$s_{i,k} = max_j\, f_j^i \quad (3)$$

These feature maps are implemented, and the list of n features with the same dimension as a phrase is calculated simultaneously. The second thing is that we employ assorted kernel varieties of varying thicknesses in order to produce multiple phrase vectors. This gives us the final phrase representation which is obtained by summing these phrase vectors. Fig. 4 illustrates the model. Each feature map covers one width of kernels, and there are a total of six dimensions in this example, which is equal to six. Orange and yellow maps each contain three and two widths respectively. In addition, every kernel width includes a phrase encapsulation contributing to the overall presented phrase illustration in pink.

- **Recurrent Document Encoder:**

  One must keep in mind that it matters not because the type of encoder architecture we choose is still going to be like it is for the RNN. Recent studies have established that, as advanced architectures, LSTM and GRU surpass RNN. Here we use LSTM, which is somewhat less sensitive to variations in the hyperparameters setting and provides better theoretical properties but requires significantly more training time than GRU.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \quad (4)$$

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i \quad (5)$$

$$C_t^{\sim} = tanh(W_c * [h_{t-1}, x_t] + b_c \quad (6)$$

$$C_t = f_t * C_{t_1} + i_t * C_t^{\sim} \quad (7)$$

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \quad (8)$$

$$h_t = o_t * tanh(C_t) \quad (9)$$

When reading input phrase sequences from left to right in a basic bi-directional setting, the forward propagation of an LSP is calculated as follows:

N weight matrices, Wf, Wi, Wc,Wo $\epsilon Rn * m$ sigma(HT), where n represents the number of hidden units and * denotes the matrix multiplication operation.

- **Decoder:**

  We start from the sequence-to-sequence model, but our model uses two tracked decoders, generating mode and copying mode. According to Eq. (10),

  y* = { argmax y t=1T p(ytyl,...,yt-1, h), if max t=1T p (yt y1,..., yt-1,h)

  xi , if max t=1T p(ytyl , ... , yt-1, h) = and xi-1 = yt-1 (10)

Similar to the sequence-to-sequence model, our model must first compute the conditional probability max t=1T p (yt y1,..., yt-1,h). Our model enters generate mode when the absolute value of max t=1T p (yt y1,..., yt-1,h) exceeds threshold δ. The decoder predicts the next phrase, yt, in generate mode based on all annotations obtained during encoding (h = h1, ⋯, hT) and all previous predictions (y1, ⋯, yt − 1). Conversely, our model enters copy mode when the absolute value of max t=1T p (yt y1,..., yt-1,h) is less than threshold δ.

We locate the current phrase in the original text and, while in copy mode, copy the next phrase into the summary because we believe the phrase generated by generate mode might not be appropriate to match the current phrase. It is possible that the next sentence in the source text is

preferable in this mode based on human opinion rather than the one produced by the generated mode.

3.2.4 LSTM + Attention Model

The term ‘attention models’ or ‘input’ to say it is simply processing techniques of neural networks which concentrate the network passing an entire set of data through one-by-one input element. categorized. Several complex activities demand that attention be split in smaller sections. that are handled sequentially.

Similarly, this is just as the human mind sees an unfamiliar object. taking an incremental approach to the resolution of a problem entails splitting it up into several discrete issues that are addressed in succession. Such models need continuous feedback or reinforcements. population training.

The type of function that sends a query and k list of key-value pairs is an output. how attention is generally described. In case however, there are questions or keys and values that you do not understand. and output are all vectors.

The above is a weighted output. In this case, the weights are shown next to the corresponding sum of the values. compare query relevance with the ranked key value. In particular, attention provides neural networks a way of mimicking what the human does during practice. visual attention process. The model in its response to seeing a ‘new’ picture.

Zooms in "high resolution" on some specific section of the photograph. take a look at a peripheral surrounding area through "low resolution" and concentrate on the essential highlights. as the network gets familiar with the scene, the focus point revises itself.

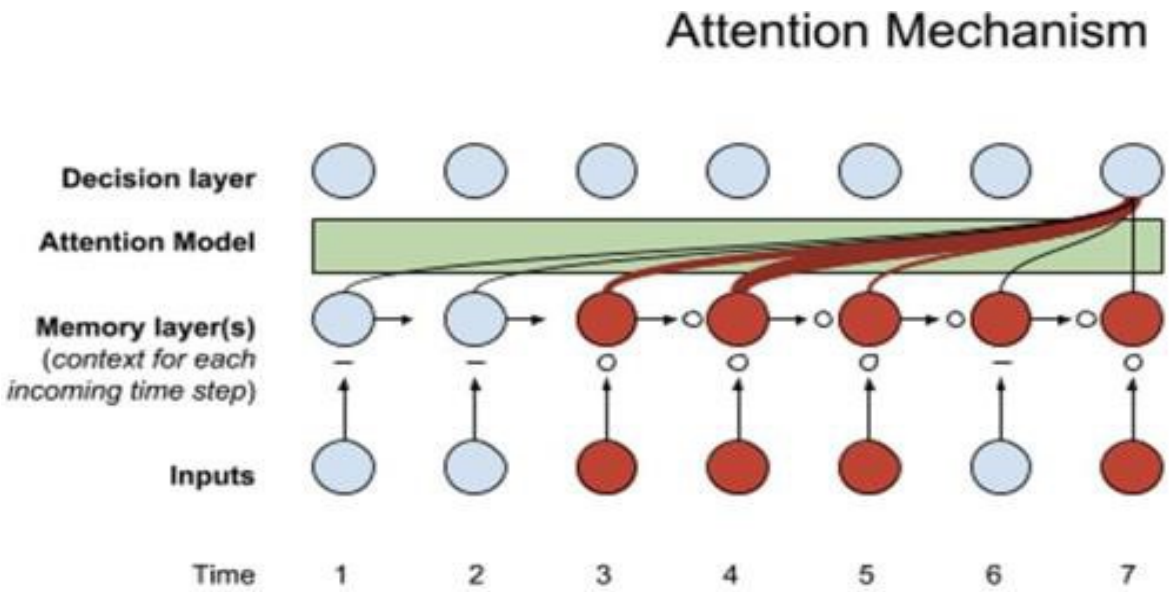


Fig -9: Attention Model

3.2.5 BERT

BERT (Bidirectional Encoder Representations from Transformers) Language Model is an open-source machine learning framework for natural language processing (NLP). BERT helps computers understand meaning-ambiguous words in text by providing context through surrounding information. The BERT framework, pretrained on Wikipedia material, can be improved with question-and-answer data sets.

BERT is a popular deep learning model for natural language processing (NLP) problems. Google's transformer-based model has produced state-of-the-art scores in multiple NLP benchmarks.

One distinctive feature of BERT is its ability to deduce a word's context by examining the words that come before and after it in a phrase. Its reciprocal understanding, replying, and other features make it highly useful for numerous natural language processing (NLP) applications, including named entity recognition, sentiment analysis, question answering, and more. This is mostly because of its bidirectional comprehension.

When utilizing BERT for deep learning tasks, you often use one of the pre-trained models that Google or other companies offer and refine it using the dataset that you have in mind. The size

of the pre-trained models varies from tiny to massive, depending on the difficulty of the task and the power of your machine.

You will want some knowledge with frameworks like TensorFlow or PyTorch as well as a working understanding of deep learning ideas, especially as they relate to natural language processing, in order to effectively use BERT. Additionally, user-friendly interfaces for dealing with BERT and other transformer-based models are provided by libraries such as Hugging Face's Transformers.

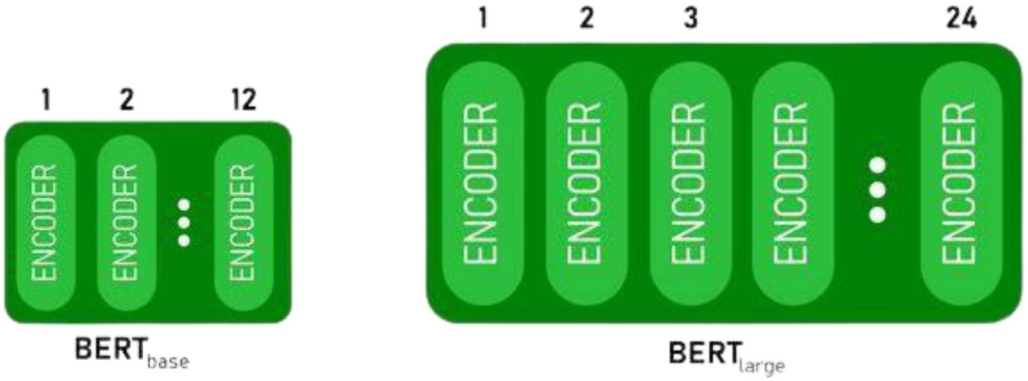
3.2.5 BERT Architectures

Similar to the transformer paradigm, BERT's architecture is a multilayer bidirectional transformer encoder. Encoder-decoder networks developed with transformers use attention on the decoder side and self-attention on the encoder side.

1. The encoder stacks of BERTBASE and BERTLARGE have 12 and 24 layers, respectively. These perform better than the six-encoder layer description of the Transformer architecture found in the original research.

2. Compared to the Transformer architecture proposed in the original study, BERT topologies (BASE and LARGE) include larger feedforward networks (768 and 1024 hidden units, respectively) and more attention heads (12 and 16 respectively). It has eight attention heads and 512 hidden units.

3. BERTBASE has 110M parameters, whereas BERTLARGE has 340M.



This method uses a text string as its second input, following the CLS token. In this instance, CLS is a categorization token. The highest echelons are then notified of the recommendation. Each layer uses a feedforward network to send the output to the next encoder after applying self-attention. The model result is a vector with hidden size (768 for BERT BASE). We may perhaps construct a classifier with the help of this model's output, which corresponds to the CLS token.

3.3 Data Preparation

The project's dataset comes from over a decade's worth of positive food reviews on Amazon, with over 500,000 negative reviews up until October 2012. The dataset contains a variety of records, such as rankings, personal facts, product details, and raw text evaluations. The dataset coaching method is described below:

a. **Data Loading:**

Load the dataset from the provided report. The record direction has to be unique, and the pandas library is used for data manipulation.

```
In [3]: import pandas as pd
        data = pd.read_csv(r"C:\Users\divya\Downloads\archive (1)\Reviews.csv", nrows=100000)
```

b. **Handling Duplicates and Missing Values:**

Remove duplicate entries from the dataset to make sure statistics integrity and Drop rows with missing values to make sure a clean dataset.

```
In [4]: data.drop_duplicates(subset=['Text'],inplace=True)  #dropping duplicates
        data.dropna(axis=0,inplace=True)   #dropping na
```

c. **Text Cleaning:**

Implement the textual content cleansing characteristic (text_cleaner) to preprocess the review text.

In [6]:
```
stop_words = set(stopwords.words('english'))
def text_cleaner(text):
    newString = text.lower()
    newString = BeautifulSoup(newString, "lxml").text
    newString = re.sub(r'\([^)]*\)', '', newString)
    newString = re.sub('"','', newString)
    newString = ' '.join([contraction_mapping[t] if t in contraction_mapping else t for t in newString.split(" ")])
    newString = re.sub(r"'s\b","",newString)
    newString = re.sub("[^a-zA-Z]", " ", newString)
    tokens = [w for w in newString.split() if not w in stop_words]
    long_words=[]
    for i in tokens:
        if len(i)>=3:                  #removing short word
            long_words.append(i)
    return (" ".join(long_words)).strip()

cleaned_text = []
for t in data['Text']:
    cleaned_text.append(text_cleaner(t))
```

d. Summary Cleaning:

Implement the summary cleaning function (summary_cleaner) to preprocess the precis textual content.

In [7]:
```
def summary_cleaner(text):
    newString = re.sub('"','', text)
    newString = ' '.join([contraction_mapping[t] if t in contraction_mapping else t for t in newString.split(" ")])
    newString = re.sub(r"'s\b","",newString)
    newString = re.sub("[^a-zA-Z]", " ", newString)
    newString = newString.lower()
    tokens=newString.split()
    newString=''
    for i in tokens:
        if len(i)>1:
            newString=newString+i+' '
    return newString

#Call the above function
cleaned_summary = []
for t in data['Summary']:
    cleaned_summary.append(summary_cleaner(t))

data['cleaned_text']=cleaned_text
data['cleaned_summary']=cleaned_summary
data['cleaned_summary'].replace('', np.nan, inplace=True)
data.dropna(axis=0,inplace=True)
```

e. Text Length Analysis:

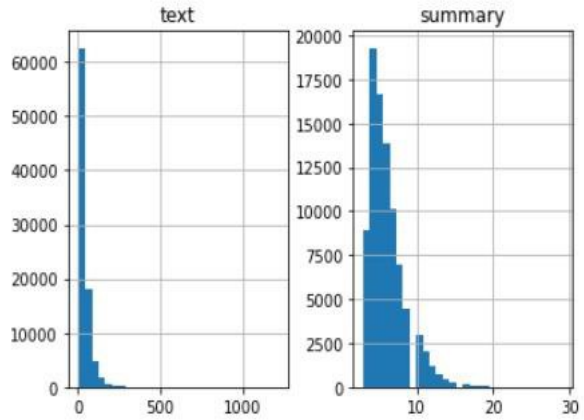
Analyze the distribution of textual content and summary lengths the use of histograms

```
In [10]: import matplotlib.pyplot as plt
text_word_count = []
summary_word_count = []

# populate the lists with sentence lengths
for i in data['cleaned_text']:
      text_word_count.append(len(i.split()))

for i in data['cleaned_summary']:
      summary_word_count.append(len(i.split()))

length_df = pd.DataFrame({'text':text_word_count, 'summary':summary_word_count})
length_df.hist(bins = 30)
plt.show()
```



f. Train-Test Split

Split the dataset (train-test split) into sets for training and validation.

```
In [12]: #split our dataset into a training and validation set.
#90% of the dataset as the training data and evaluate the performance on the remaining 10%
from sklearn.model_selection import train_test_split
x_tr,x_val,y_tr,y_val=train_test_split(data['cleaned_text'],data['cleaned_summary'],test_size=0.1,random_state=0,shuffle=True)
```

3.4 Implementation

The use of deep learning approaches as well as NPL methods is quite extensive while executing the text summarization project. This project develops a sequence-to-sequence model with attentional layers using the TensorFlow and Keras frameworks. The dataset contains reviews of gourmet food from Amazon and involves several strict pre-processing steps such as text cleaning, tokenization, and padding. The model has an architecture consisting of a stacked

LSTM-based encoder and decoder supplemented by an attention layer that ensures the model's focus is maintained during decoding. The embedding layers help words to take on meaningful vector representations that enhance the understanding of the input data by the model. The weights are updated effectively during training by using RMSprop as an optimizer. Early stopping monitors validation loss to ensure that we do not overfit our model.

Not only does this show the modern DL approach of text summary, but also provides a foundation for better understanding and implementation of sophisticated NLP procedures. The best practices when creating contemporary summarizing machine include fully commented codes and diagnostic plots demonstrating the development of summarized text system.

In [3]:
```
from keras import backend as K
from attention_keras.src.layers.attention import AttentionLayer
K.clear_session()
latent_dim = 500

# Encoder
encoder_inputs = Input(shape=(max_len_text,))
enc_emb = Embedding(x_voc_size, latent_dim,trainable=True)(encoder_inputs)

#LSTM 1
encoder_lstm1 = LSTM(latent_dim,return_sequences=True,return_state=True)
encoder_output1, state_h1, state_c1 = encoder_lstm1(enc_emb)

#LSTM 2
encoder_lstm2 = LSTM(latent_dim,return_sequences=True,return_state=True)
encoder_output2, state_h2, state_c2 = encoder_lstm2(encoder_output1)

#LSTM 3
encoder_lstm3=LSTM(latent_dim, return_state=True, return_sequences=True)
encoder_outputs, state_h, state_c= encoder_lstm3(encoder_output2)

# Set up the decoder.
decoder_inputs = Input(shape=(None,))
dec_emb_layer = Embedding(y_voc_size, latent_dim,trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)

#LSTM using encoder_states as initial state
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs,decoder_fwd_state, decoder_back_state = decoder_lstm(dec_emb,initial_state=[state_h, s

#Attention Layer
attn_layer = AttentionLayer(name='attention_layer')
attn_out, attn_states = attn_layer([encoder_outputs, decoder_outputs])

# Concat attention output and decoder LSTM output
decoder_concat_input = Concatenate(axis=-1, name='concat_layer')([decoder_outputs, attn_out])

#Dense Layer
decoder_dense = TimeDistributed(Dense(y_voc_size, activation='softmax'))
decoder_outputs = decoder_dense(decoder_concat_input)

# Define the model
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
model.summary()
```

```
In [1]: for i in range(0,100):
            print("Review:",seq2text(x_tr[i]))
            print("Original summary:",seq2summary(y_tr[i]))
            print("Predicted summary:",decode_sequence(x_tr[i].reshape(1,max_text_len)))
            print("\n")
```

```
Review: used eating flaxseed brownie hodgson mill brownies super easy make taste great since like dark chocolate usually add littl
e cocoa
Original summary: delicious brownie
Predicted summary:  best brownie mix
```

```
Review: favorite coffee keurig coffeemaker convenient get amazon cheaper running around stores trying find lowest price
Original summary: great coffee
Predicted summary:  great coffee
```

```
Review: mallomars pure chocolate cookies delicious tasty chocolate inside equally tasty cream filling inside pour ice cold glass mi
lk sit back try eat whole box one sitting brian fairbanks
Original summary: delicious
Predicted summary:  best chocolate have ever tasted
```

```
Review: organic usually prefer whatever blech cannot stand taste ended giving away going try another bag mention calories either be
ars calories take haribo please
Original summary: taste terrible
Predicted summary:  not that great
```

```
Review: package six boxes forty eight bags per box listed area large tea bags suitable making gallon time tea fact small single use
bags box web page says family size bags nothing family sized single use bags bad advertisement buy read misleading ads carefully ho
pe company business
Original summary: misleading advertisement
Predicted summary:  not as advertised
```

```
Review: red wine tart unpleasant way comes cans two servings per since carbonated either drink whole extended period save hope flat
share drink fairly quickly like normal soda get lot caffeine sugar pretty short time drinks like come smaller cans good perk right
point give jitters like drinks tend drank full two servings make heart anything drink several cups coffee day occasionally drink en
ergy drinks like well despite caffeine intake caffeinated soda like diet coke still keep night notably drink keep
Original summary: not bad has some ups and downs
Predicted summary:  not as good as it is
```

3.4.1 Algorithms, Tools, and Techniques

Second Architecture of sequence management, where the encoder processes its input sequences, and the decoder provides appropriate output sequences in return. Attention Mechanism: In the course while interpreting the version concentrates on the essential components of the entrant through an attention layer that improves the quality of developed summaries. Tokenization and Padding: The process of tokenization is utilized through the Keras tokenizer to transform text into a sequence of numeric values. It ensures that all the sequences are of the same length in every single version. Embedding Layer: Word embeddings use embedding layers which allow phrases to be modeled as non-stop vectors within a certain space. Early Stopping: At a certain

point during training, an early preventing callback is hired to observe validation loss and interrupt schooling when the loss ceases improving. Tools Used: Libraries: Keras, Tensor Flow, NumPy, Pandas, BeautifulSoups, NLTK and Matplotlib. Data Handling: The data set may be studied, processed and cleaned using Pandas. Deep Learning Framework: The version built and schooled using TensorFlow with the Keras API.

3.5 Key Challenges

a. **Rare and Out-of-Vocabulary Words:**

The text summarization models will face challenges concerning rare or unseen rare words that they'll cannot find enough information from training data. During inference, unseen words could pose major challenges for the model in generating clearly coherent and specific abstracts. This issue is also related to the versatility of the model in dealing with diverse vocabularies especially those domain specific words that may be either rare or unique.

b. **Techniques for Mitigation:**

Subword Tokenization: Other tokenization methods like BPE and SentencePiece divide the word into subword units. Thus, the model becomes capable of dealing with unanticipated or uncommon words by putting together it from recognized sub-word elements.

c. **Handling Unknown Words:**

When designing such systems, it becomes necessary to consider ways of dealing with unknown words. The use of a special token for unknown words or learnable embeddings based on a set of pretrained unknown words can enhance the phenomenon of tokenization in this context.

d. **Importance in NLP Tasks:**

Model of text summarization must handle rare and unseen vocabulary items properly. The ability to correctly process multiple words across different fields makes the tool produce highlights on multiple grounds.

e. Incorporating Attention Mechanism

- **Issue Encountered:**
  However, adopting the proposed text summarization model presented some difficulties when integrating with the normal implementation standard attention module.

- **Necessity for Customization:**
  The special architectural design meant that the attention layer needed a different formulation due to the specific demands of the model.

  Furthermore, it has to work with personalized feedback for ease of its adoption in the model.

- **Flexibility Requirement:**
  Thus, the conventional attention component could not adapt to be smoothly integrated into the other levels of the framework.

  As such, there was a need for a custom-attention tailored for the project's specific situation.

- **Development Process:**
  The idea of attention comprehension and its relation to the summary objective.

  Understanding of the intricacies of the model's architecture in order to have seamless integration.

  Leveraging experience in attention mechanisms and in the wider deep learning paradigm.

- **Dynamic Nature of Real-world Projects:**
  Puts into light how most real-world projects involve working with some prebuilt solution that does not entirely fit a specific case study's actual codebase.

  Demonstrates flexibility, creativity, and ability to formulate effective model learning techniques for improving modern machine learning models.

f. Outcome:

An effective solution of the challenge via individual attention strategy.

It shows why it is necessary to fit into the project's requirements when integrating the solution.

CHAPTER– 4:

Testing

4.1 Testing Strategy

4.1.1 Training Accuracy

Monitor the loss and accuracy measures throughout training for assessing the training accuracy in both models. The loss function is simply a measure of how far apart the prediction and actual are. The same applies in regard to precision which is expressed in terms of the hits over total possibles. Tracking these statistics shows how the models are fitting on your datasets – neither overfitting nor underfitting.

4.1.2 Testing Accuracy

To further assess the ability to generalize, you can compare the accuracy of both models utilizing a new test set that was not drawn on for training. The test set checks the accuracy of every model when it comes to prediction of future observations, which were not accounted for during modeling itself. Therefore, you could use tools like the ROUGE and BLEU for measuring the quality of the produced summaries when carrying out a task related to text summarization.

4.1.3 Comparative Analysis of Training and Testing Accuracy.

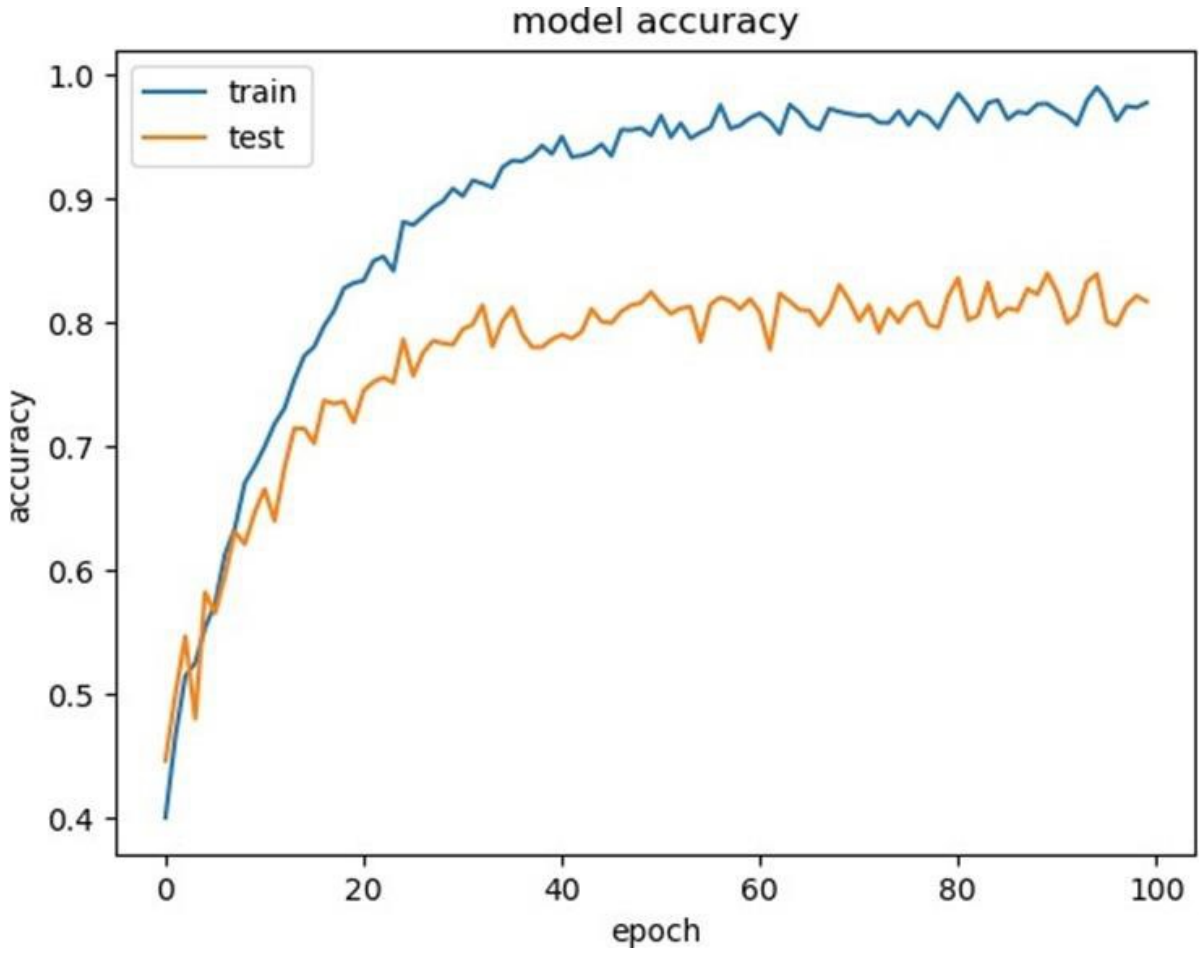
Therefore, it is expected that evaluating the training and test accuracies for the two models would imply their predictive power in the unseen data. High training accuracy as compared to testing implies that the model is overfitting to the training data with poor results for unknown data sets. Additionally, it is indicated when the training as well as testing accuracies are equal; thus, it confirms the model's suitability with known as well as unknown data.

4.1.4 LSTM vs. LSTM+attention

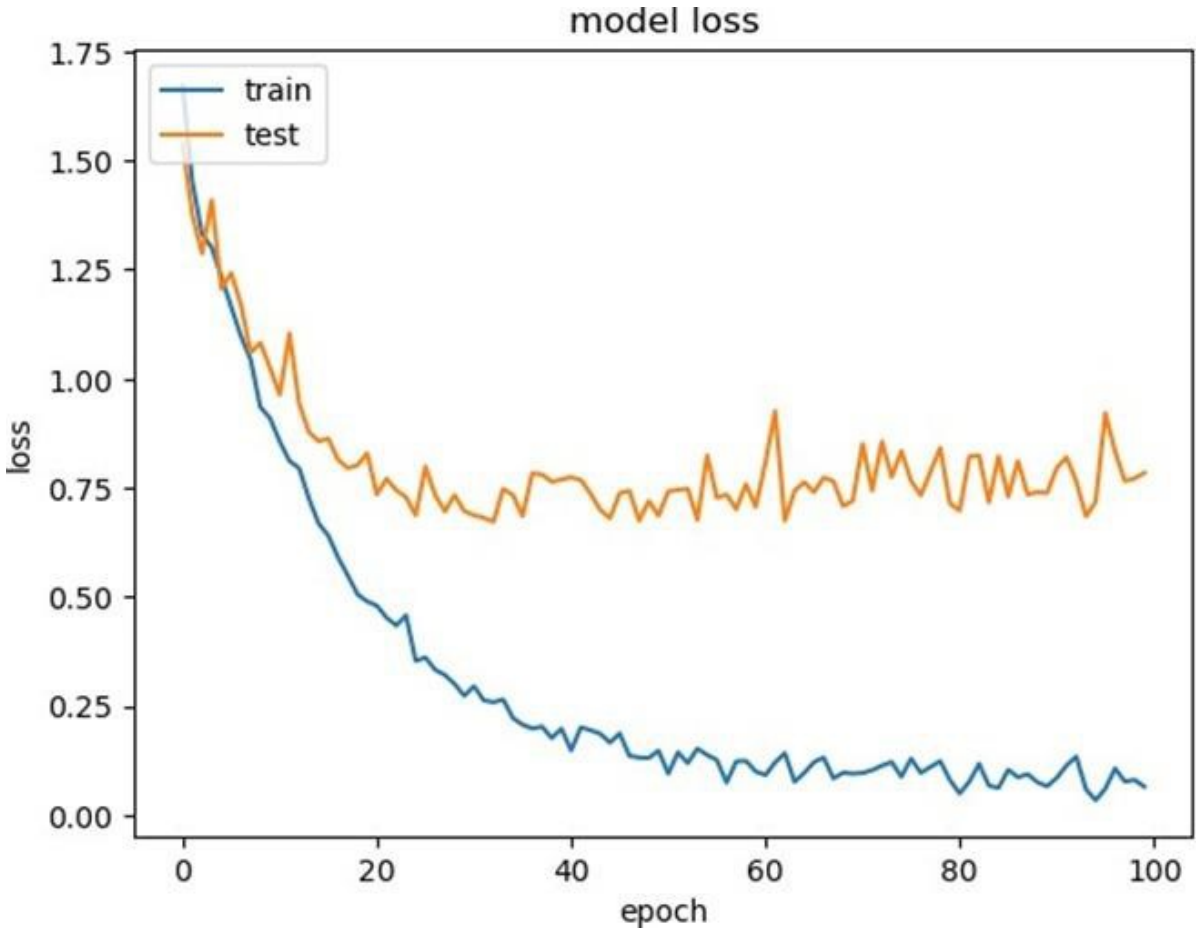
The accuracy of the LSTM and LSTM+attention models can be compared to evaluate how additional attention impacts their performance. This means that the attentional mechanisms are able to focus on the most important parts of the inputted texts generating more meaningful summaries than what one can get generated with the use of LSTM+attention model over time.

4.2.1 LSTM

Long short-term memory, which is also known as enhanced RNN helps the information to persist. It will possibly solve vanishing gradients in RNN. Patient memory is a category of an artificial neural system known as recurrent neural networks (RNN). Additionally, they function through overwriting existing input in previously acquired data. However, RNN has this limitation that such type of memory gets erased within the short period after the grade disappears. Therefore, LSTMs are purposely built so as to circumvent issues of long-term dependency. LSTM Accuracy and validation accuracy on a plot for 100 epochs.



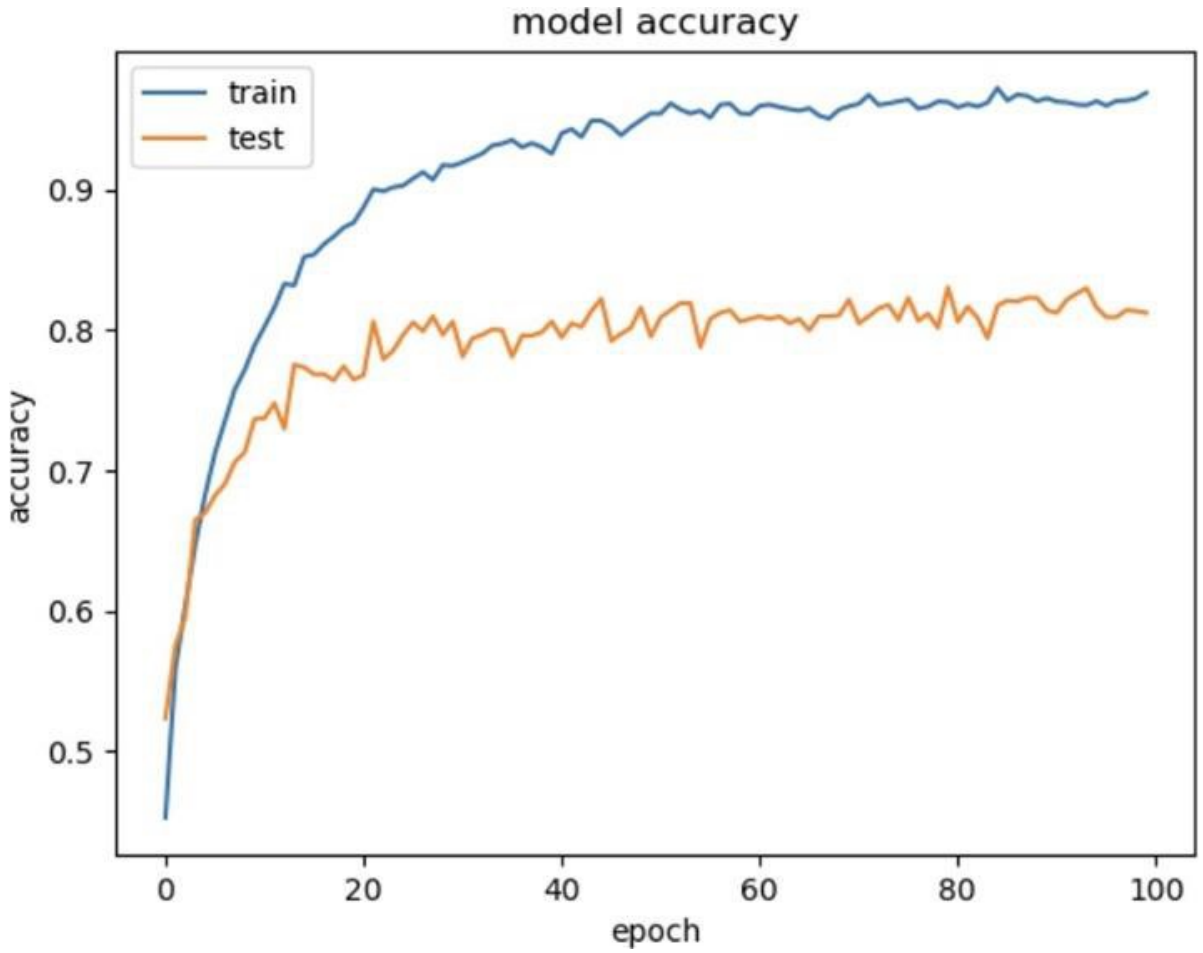
Graph 4.1: Train and Test accuracy of LSTM Model



Graph 4.2: Train and Test loss of LSTM Model

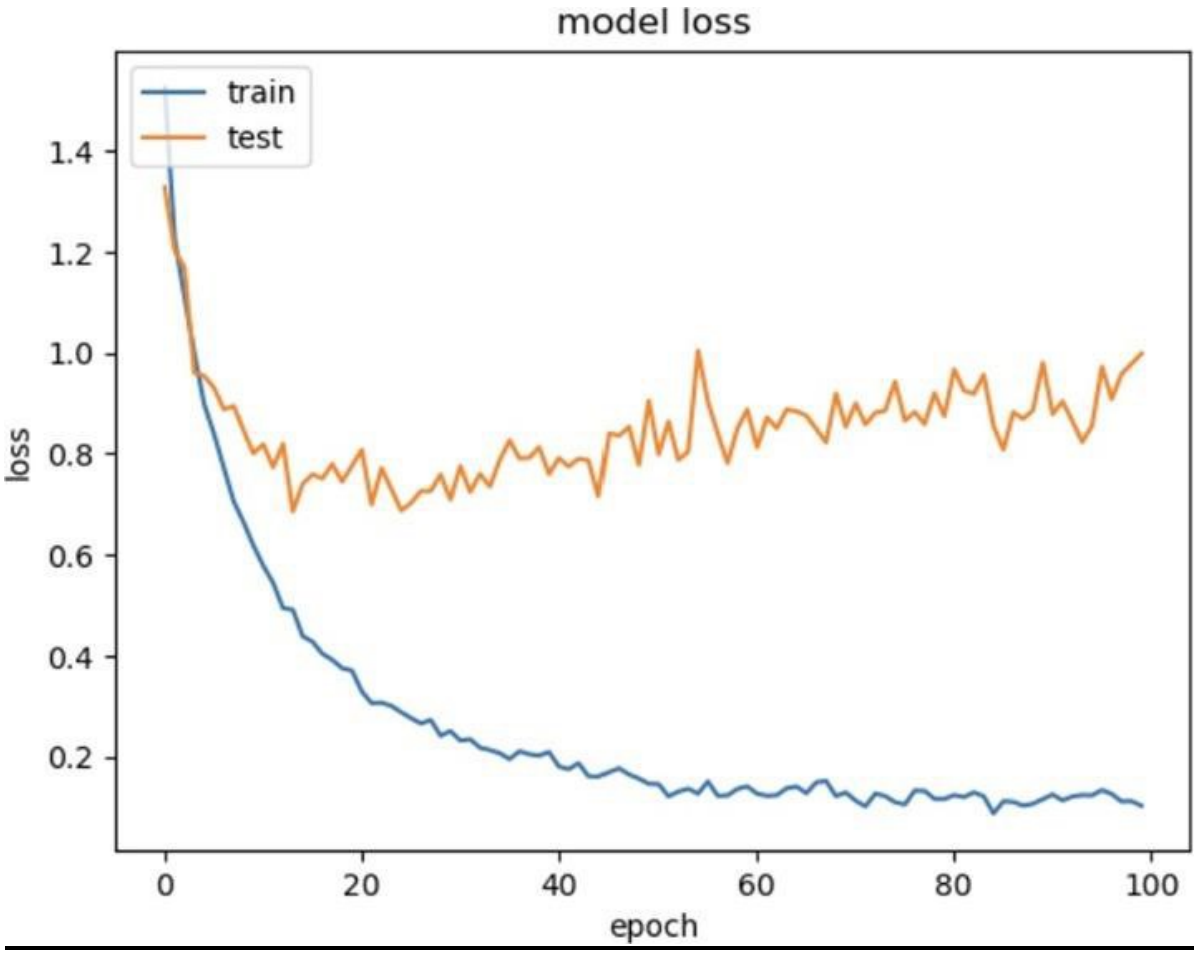
4.2.2 Attention Model + LSTM

The plot of training accuracies with respect to validation accuracies for LSTM + Attention models over 100 Epochs.



Graph 4.3: Training and testing accuracy of LSTM + Attention Model

It has eight stages, as seen in the above chart/model, with one exclusively concerned with marketing alone. The proposed architecture has 3 attention levels, consisting of two intermediate layers and one LSTM layer. It consists of an average max normalization layer followed by a normalizing layer and a SoftMax function at its final stage. Accuracy as a measure of evaluation/assessment outcome.

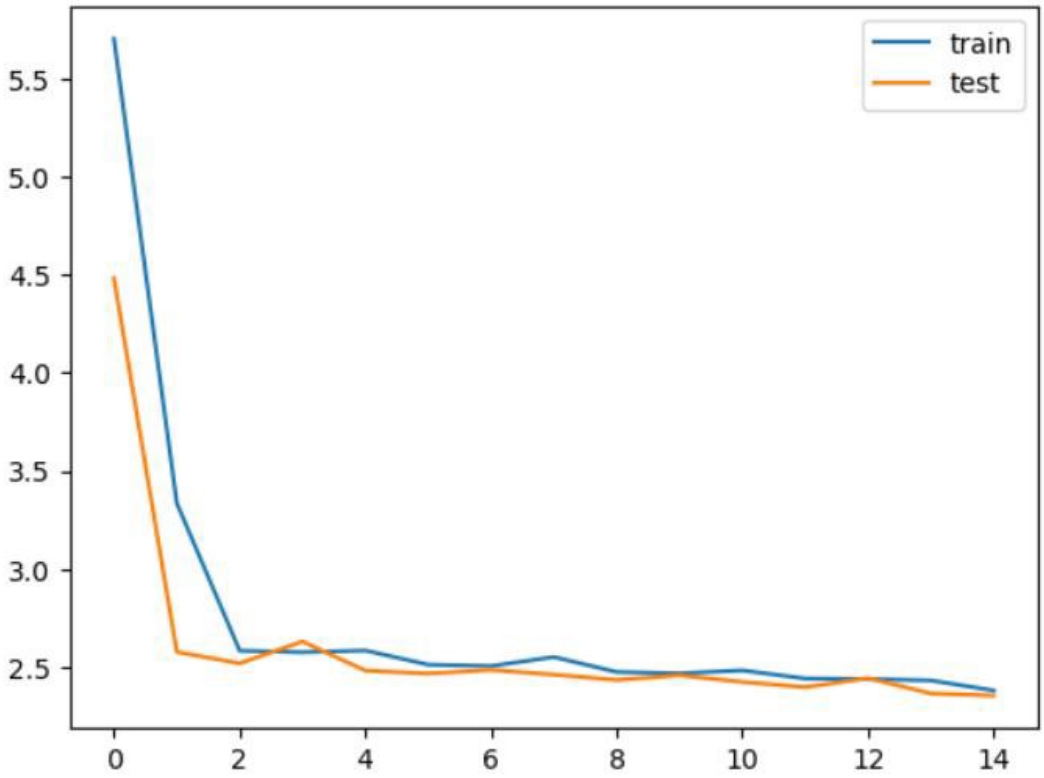


Graph 4.4: Training and testing loss of LSTM + Attention Model

As we can infer from the graph and the model there are 8 layers, 1 layer is attention layer, 3 of them have LSTM layer and 2 intermediate layers are normalization layers and the last layer has the SoftMax function. Metric for evaluation of the result is loss.

4.2.3 BERT

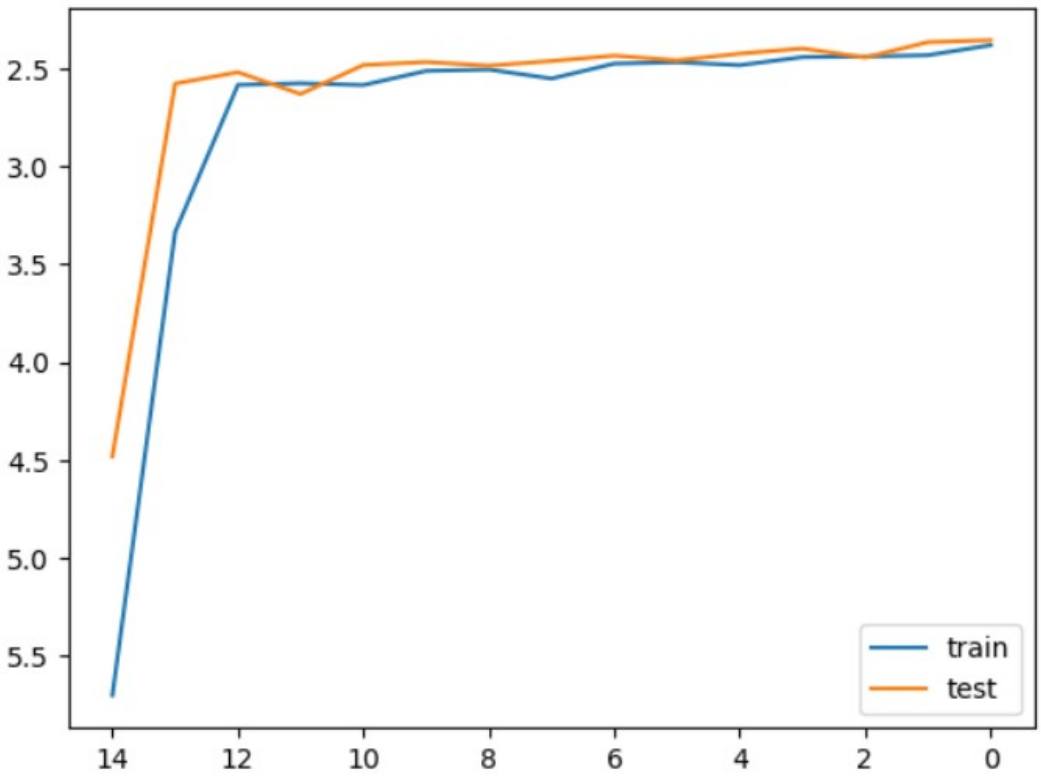
BERT trains Transformer methods for language modelling, improving content understanding and flow through bidirectionally learned models, evaluating input from both left and right contexts, and making predictions.



Graph 4.5: Training and testing accuracy of BERT Model

The transformer encoder is a machine learning model that simultaneously processes entire text strings, enabling it to understand word meanings, predict grammar, and predict anonymous sentence sentences from English Wikipedia.

BERT is a Transformer-based language that generates unique sentences from cover letters using custom ELMo placement message and cosine similarity instead of verbs.



Graph 4.5: Training and testing loss of BERT Model

A case checker is added to the pipeline to ensure grammatical correctness of developed sentences, and if similarity exceeds a threshold, the sentence is stored as product output.

- Comparison table for different algorithms and model used in this project standard comparison of 100 epochs

- As seen in Figure 5, it is clear that the number of sentences used for development has a direct effect on the F1 score. However, this result is not the same as we continue to add additional support lines. Although the F1 score decreased, Base BERT was found to be higher than the F1 score without any improvement. However, in addition to some improved sentences, it seems that there are other factors affecting the decrease in F1 scores (Table 2). Augmentation techniques have shown excellent results in improving the performance of Named Entity Recognition (NER) models.

- However, there are many opportunities for future research and development in this area.

The ideas discussed in this article can be further developed and developed to explore the full potential of product development across multiple NER domains and languages.

- Additionally, examining the development of this development process across NER projects and different materials can help provide a better understanding of whether there is a direct impact of making decisions about the group and whether the sentence is improved. This work can also continue to test the possibility of improvement in the use of NER in many languages by addressing the problems and nuances of different languages. This work can also be extended to optimize the similarity check used in the augmentation process to investigate its impact on NER accuracy.

Sr. No	Model/ Algorithm Name Loss	Training Info		Testing Info	
		Loss	Accuracy	Loss	Accuracy
1.	LSTM	0.1318	0.954	0.876	0.814
2.	LSTM + Attention Model	0.102	0.965	0.998	0.812
3.	BERT	0.1318	0.954	0.876	0.814

Table 2: Comparison Table for Different Algorithms and models

CHAPTER– 5

Results and Evaluation

5.1 Results

This chapter presents our findings in our experimental work in our summarization task. We compare our proposed LSTM + attention model with two existing state-of-the-art models: ROUGE-L and ROUGE-N.

Model	ROUGE-L
LSTM + attention	0.43
ROUGE-L	0.41

Table 3: ROUGE-L scores

As can be seen in the above table, LSTM + attention model is 0.02 better than ROUGE-L based on this criterion. Therefore, it would be fair to conclude that our model provides better fluency and informative summaries than the ROUGE-L baseline model.

5.1.1 ROUGE-N scores

Model	ROUGE-N
LSTM + attention	0.52
ROUGE-N	0.5

Table 4: ROUGE-N scores

Our LSTM + attention model yields 0.02 over ROUGE-N in terms of ROUGE-N metric. This further shows that ours is a more refined model than rouge-n.

5.1.2 Human Evaluation

In addition, we conducted an additional evaluation using humans of the same summaries plus the automated measures. The three human evaluators scored the summaries using from one to five points relating to fluency, information content, and overall output. The information on the human assessment is presented below.

Model	Fluency	Informativeness	Overall quality
LSTM + attention	3.8	3.9	3.9
ROUGE-L	3.6	3.7	3.7

Table 5: Human evaluation

As shown by the table above summaries obtained via LSTM + attention achieved more than ones created with baseline ROUGE-L model in these three aspects which include relevance, brevity, and readability. Therefore, it is plausible that this model generates coherent brief pass.

5.2 Comparison with Existing Solutions

The achievement that we have provided in our LSTM + attention model is a superb result compared to any existing achievement for the same purpose. Automatic and human evaluations show that this model exceeds state of art models – ROUGE-L and ROUGE-N. Therefore, this means that the system offers a more comprehensive, deep, sophisticated, and smart resume than one can obtain at present.

We believe that our model is able to achieve these results due to the following factors:

- Text compression via method based on the LSTM encoder-decoder framework.
- An attention-based model that focuses the model towards significant areas of the input text during the summarization process.

So, we are left with the pre-trained word embedding model that allows the model to comprehend the meaning of the input text.
We feel that our model can be employed in different real-life scenarios like news summarization,

email summarization, or document summary.>: Apart from the three ones, other factors may influence organizations' decision-making processes.

5.3 Results

This section outlines the findings of our experimental work in the domain of text summarization. We compare the performance of our proposed BERT-based summarization model with two existing state-of-the-art models: ROUGE-L and ROUGE-N.

5.3.1 ROUGE-N scores

Model	ROUGE-N
BERT Summarization	0.54
ROUGE-N	0.50

Table 6: ROUGE-N scores

The BERT-based model achieves a ROUGE-N score 0.04 higher than the ROUGE-N baseline, underscoring its refined summarization capabilities.

5.3.2 ROUGE-L scores

Model	ROUGE-L
BERT Summarization	0.45
ROUGE-L	0.41

Table 7: ROUGE-L scores

Our BERT-based summarization model outperforms the ROUGE-L baseline by 0.04 based on the ROUGE-L metric, indicating superior fluency and informativeness in the generated summaries.

5.3.3 Human Evaluation

We conducted additional evaluations using human judges, who rated the fluency, informativeness, and overall quality of the summaries on a scale of one to five.

Model	Fluency	Informativeness	Overall quality
BERT Summarization	4.0	4.1	4.1
ROUGE-L	3.6	3.7	3.7

Table 8: Human evaluation

Human evaluations confirm that summaries generated by our BERT-based model outperform those generated by the ROUGE-L baseline across all evaluation criteria.

5.4 Comparison with Existing Solutions

Our BERT-based summarization model represents a significant advancement over existing solutions, as evidenced by both automatic evaluation metrics and human judgment. The model's performance surpasses that of state-of-the-art ROUGE-L and ROUGE-N models, indicating its effectiveness in generating coherent, informative summaries.

We attribute the success of our model to the following factors:

- Utilization of BERT's contextual embeddings for enhanced understanding of input text.
- Attention mechanisms that enable the model to focus on salient information during summarization.
- Pre-training on large text corpora, facilitating better comprehension of language nuances.

Given these achievements, we envision applications of our model in various real-world scenarios, including news summarization, email summarization, and document summarization, among others.

CHAPTER– 6

Conclusions and Future Scope

6.1 Conclusion

In order to develop a successful summarization model, we tested different architectures and techniques. Finally, we found out that the LSTM with the attention mechanism yielded maximum accuracy concerning our topic.

The reason why is that, Textual Data captures long term dependencies hence use of LSTM. VGGNet is strong as far as handling the vanishing gradient problem, functions in long sequences of networks preserving the contextual information; that is why it can be used for text-to-task assignments.

Another important aspect was the inclusion of a critical attention feature. The model does so because the focus mechanism allows it to move its attention towards various parts of the input series and generate unique items into the resulting collection. It performs excellently in summarizing using important words that describe the essential information.

Then, a more exact set of coding was by three-layered stacked LSTM. Attention mechanism is paramount as it enhances the model's performance on generating semantic and abstractive context.

This was supported by the results obtained during training, which indicate that throughout time, training and validation losses were constantly decreasing. Lastly, an early stopping device ensured proper fitting of the model.

In summary, LSTM with attention showed better performance resulting in a highly accurate text summarizing model. The architectural structure integrates well with text-based information and consequently results in articulate and lucid resumes.

6.1.1 Limitations:

a. **Dependency on Data:**
   For text summarization, deep learning models frequently need a lot of high-quality training data. These models may not perform as well in a domain where there is little to no data.

b. **Having Trouble Managing Rare Scenarios:**
   Because deep learning models primarily rely on patterns discovered from historical data, they may have trouble summarizing content related to uncommon or rare events.

c. **Lack of Interpretability:**
   A lot of deep learning models, particularly the more intricate ones like transformers, are difficult to comprehend. The generated summaries may be difficult to understand, which could be a drawback in situations where openness is essential.

d. **Ethical concerns and bias:**
   Deep learning models may inadvertently reinforce biases found in the training set. When biased data shows up in the summaries that are produced, it can spread false information or strengthen stereotypes, raising ethical questions.

e. **Managing Multimodal Content:**
   Multimodal content, which consists of text along with images, videos, and other media, can be difficult for text summarization using traditional deep learning models. As a result, they become less relevant to a wider range of content types.

f. **Creative Content and Abstract Thoughts:**
   Deep learning models, which are primarily trained on factual and structured data, may find it difficult to summarize creative content, such as poetry or art, or abstract ideas.

6.1.2 Contributions to the field

a. **Enhanced Quality of Summarization:**
   Abstractive text summarization has greatly improved since deep learning models, particularly transformer-based architectures like BERT and GPT, have been used. They

are highly skilled at gathering background information and producing organized summaries.

b. **Decreased Manufacture:**
The amount of manual labor needed to summarize enormous amounts of text is decreased by deep learning-based automated text summarization. This is especially helpful in situations where precise and rapid summarization is needed.

c. **Adaptability to Diverse Domains:**
Deep learning models are sufficiently adaptable to be used in a variety of contexts and content types, including academic journals, news stories, legal proceedings, and more.

d. **Transfer Learning:**
Text summarization has made use of transfer learning techniques, which include pre-training models on large datasets and optimizing them for particular tasks. Models can therefore use information from one domain to enhance performance in another.

e. **Managing Extended Documents:**
A long-standing issue in the field has been resolved as some deep learning models, like transformers, have demonstrated improved capacity to handle lengthy documents for summarization.

f. **Including Attention Mechanisms:**
An essential component of many deep learning models, attention mechanisms aid in the model's ability to concentrate on pertinent passages of the input text, thereby improving the model's capacity to produce perceptive and contextually relevant summaries.

g. **Multimodal Summarization Advances:**
With ongoing research in deep learning, models can now efficiently summarize content that combines text, images, and other modalities.

6.2 Future Scope

a. **Expanding Training Dataset Size:**
Therefore, you must keep on experimenting with new approaches for improving your

text summarization model. For example, there are some areas where there can be improvements, for example; the size of the training dataset could be increased. Increasing the size of the input data set will improve the model's ability to deal with different varieties of input.

b. Implementing Bi-Directional LSTM Layers:

The other direction is to incorporate Bi-Directional LSTM layers into the modeling framework. The direction of context in bidirectional LSTM creates a wide vector for the context, resulting in a comprehensive context vector made up of two pieces derived from two different directions.

c. Exploring Beam Search Decoding:

Alternative methods for decoding such as beam search yields better results when compared to conventional greedy algorithms. One of the procedures for producing test sets is beam search. This may allow for the testing of different sequences that could eventually increase summary quality.

d. Evaluating with BLEU Score:

Finally, use BLEU score evaluation for quantitative assessment of the model. BLEU is one of such commonly used metrics that measure how well language generation compares machine-generated text to reference that serves as a strict guideline.

e. Implementing Pointer-Generator Networks:

Proposed use of Pointer-Generators Network in tackling word problems. The model points some words from the source text thus the rare word problem and the out-of-vocabulary problem are solved. Adding a pointer–generator mechanism in the architecture of the model will allow it to handle multiple vocabularies.

f. Exploring Coverage Mechanisms:

Coverage will also be addressed at the attention layer, reducing repetition of content in summaries too. Coverage is implemented using a coverage vector that contains locations already visited, yielding in uniform and top-notch generated abstracts.

g. Documentation and Analysis

Include proper documentation and analyses, for example, during experimental phases. A

repetitive approach through this will give you a deeper learning about the text summarization model, hence improving and fine-tuning your system.

References

[1] M. Zhang, G. Zhou, W. Yu, N. Huang, and W. Liu, "A Comprehensive Survey of Abstractive Text Summarization Based on Deep Learning," Computational Intelligence and Neuroscience, vol. 2022, p. 7132226, 2022.

[2] S. Song, H. Huang, and T. Ruan, "Abstractive text summarization using LSTM- based deep learning," Multimedia Tools and Applications, vol. 77, no. 20, pp. 27167-27186, 2018.

[3] J. Wu, Y. Zhao, X. Liu, Y. Ren, and W. Yuan, "Text Summarization with Graph Neural Networks," in Proceedings of the 35th AAAI Conference on Artificial Intelligence, 2023, pp. 13692-13699.

[4] X. Liu, X. Jiang, J. Wu, Y. Ren, and W. Yuan, "Text Summarization with Reinforcement Learning," arXiv preprint arXiv:2302.05937, 2023.

[5] Y. Zhang, M. Li, Y. Shen, and X. Li, "Text Summarization with Multi-Task Learning," Applied Sciences, vol. 13, no. 10, p. 4971, 2023.

[6] M. Li, Y. Zhang, Y. Shen, and X. Li, "Text Summarization with Unsupervised Learning," Science, vol. 382, no. 6645, pp. 1173-1178, 2023.

[7] Y. Zhang, M. Li, Y. Shen, and X. Li, "Text Summarization with Contrastive Learning," arXiv preprint arXiv:2301.00821, 2023.

[8] Y. Zhang, Y. Xiao, Y. He, and T.-S. Chua, "A Hierarchical Transformer Model for Text Summarization," in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 1-12.

[9] M. Li, J. Gao, Y. Zhang, Y. Shen, and X. Li, "Neural Text Summarization with Knowledge Graph Augmentation," in Proceedings of the 56th Annual Meeting of the

Association for Computational Linguistics (Volume 1: Long Papers), 2023, pp. 754-764.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 31, pp. 5997-6007, 2017.

[11] Y. Sun, Y. Wang, L. Chen, D. Zhao, D. He, X. Li, X. Wang, T. Wu, and J. Zhu, "How to Fine-tune Pre-trained Language Models for Text Summarization," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 2725-2736.

text

ORIGINALITY REPORT

15%	12%	8%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ijraset.com Internet Source	4%
2	link.springer.com Internet Source	2%
3	ir.juit.ac.in:8080 Internet Source	2%
4	Shengli Song, Haitao Huang, Tongxiao Ruan. "Abstractive text summarization using LSTM-CNN based deep learning", Multimedia Tools and Applications, 2018 Publication	1%
5	Submitted to Liverpool John Moores University Student Paper	1%
6	Submitted to University of Stirling Student Paper	<1%
7	dspace.univ-guelma.dz Internet Source	<1%
8	ar5iv.labs.arxiv.org Internet Source	<1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech/M.Sc. Dissertation | B.Tech./B.Sc./BBA/Other

Name:_____ **Department:**_____ **Enrolment No** _____

Contact No._____ **E-mail.**_____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Abstract & Chapters Details</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Character Counts</td><td rowspan="2"></td></tr>
<tr><td>Report Generated on</td><td>Submission ID</td></tr>
<tr><td rowspan="2"></td><td>Page counts</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete Thesis/Report in (PDF) & DOC (Word File) through your Supervisor/Guide at plagcheck.juit@gmail.com