TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

Final Year Project Report on

Abstractive Text Summarization using Transformer Model

For the partial fulfillment of the requirements of the degree of

Bachelor of Science in Computer Science and Information Technology

(B.Sc.CSIT)

awarded by Tribhuvan University

Submitted By:

Bibhav K.C. (T.U Roll No:20325/075)

Pratik Jung Karki (T.U Roll No:20343/075)

ST. XAVIER'S COLLEGE

Maitighar, Kathmandu, Nepal

Submitted To:

Office of the Dean

Institute of Science and Technology

Tribhuvan University

Kathmandu

10th March 2023

ST. XAVIER'S COLLEGE
DEDICATED TO EXCELLENCE, LEADERSHIP, SERVICE
A HIGHER EDUCATION INSTITUTE RUN BY THE NEPAL JESUIT SOCIETY
| MAITIGHAR, KATHMANDU, NEPAL | www.sxc.edu.np |
ESTD: 1988

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the Department of Computer Science, St. Xavier's College for acceptance, a Final year Project Report entitled **"Abstractive Text Summarization using Transformer Model"** submitted by **Bibhav K.C (T.U Roll No: 20325/075)** and **Pratik Jung Karki (T.U Roll No: 20343/075)** for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University.

……………………………..

Mr. Jeetendra Manandhar,
Project Supervisor,
Administrator,
St. Xavier's College

....………………………………

Mr. Ganesh Yogi,
Internal Examiner/HoD,
Department of Computer Science,
St. Xavier's College

……………………………….

Mr. Bikash Balami
External Examiner
CDCSIT, TU

DEPARTMENT OF COMPUTER SCIENCE
ST. XAVIER'S COLLEGE, MAITIGHAR, KATHMANDU

SUPERVISOR RECOMMENDATION

This is to certify that the final year project entitled **"Abstractive Text Summarization using Transformer Model"** is an academic work completed by **Bibhav K.C (T.U Roll No: 20325/075)** and **Pratik Jung Karki (T.U Roll No: 20343/075)** submitted in the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Institute of Science and Technology, Tribhuvan University under my guidance and supervision.

The information presented by him/her in the project report has not been submitted earlier to the best of my knowledge,

……………………………..
Signature of the Supervisor
Mr. Jeetendra Manandhar,
Administrator,
St. Xavier's College
10th March 2023

ACKNOWLEDGEMENT

We are immensely grateful for the privilege of being students of Computer Science at St. Xavier's College, where we have been surrounded by a department brimming with expertise and wholly supportive of its learners. We extend our sincerest gratitude to our supervisor, **Mr. Jeetendra Manandhar, Administrator, St. Xavier's College** for creating an environment conducive to both academic and social growth, and for providing us with valuable guidance, advice, and resources that have been instrumental in the successful completion of this project.

We also express our deep appreciation to the entire Computer Science Department of St. Xavier's College for providing us with a suitable working environment and for their unwavering support throughout this project. We are grateful for their guidance and assistance, which have helped us in every possible way. We would like to extend a special note of thanks to **Mr. Ganesh Yogi**, **Head of Department, St. Xavier's College**, for his continuous encouragement and support, which have been invaluable in our efforts.

In the end, we would like to express our sincere thanks to all our friends and others who helped us directly or indirectly during this project work.

Bibhav K.C (T.U. Exam Roll No. 20325/075)
Pratik Jung Karki (T.U. Exam Roll No. 20343/075)

ABSTRACT

Abstractive text summarization using transformer models has emerged as a promising approach to generating concise and meaningful summaries of large text documents. Transformer models, like BERT and GPT, are already trained language models that perform at the cutting edge on a variety of text summarization tasks. Unlike extractive summarization techniques, which select and concatenate important sentences from the input text, abstractive summarization using transformer models generates summaries by generating new text that captures the most relevant information from the original text. This approach allows for more natural and fluent summaries that are closer to human-like summaries. However, challenges such as model complexity, data scarcity, and maintaining coherence in generated summaries remain. Nevertheless, the growing body of research in abstract text summarization using transformer models is a testament to their potential and promises to lead to further improvements in this field. Abstractive text summarization using Pegasus has been applied to a variety of domains, including news articles, scientific papers, and social media. Despite its effectiveness, challenges remain, such as the difficulty in capturing the nuances of the original text and the need for large amounts of high-quality training data. Overall, abstractive text summarization using Pegasus has significant potential to improve the efficiency and effectiveness of summarizing large volumes of text in various domains.

Keywords NLP, Pegasus, abstractive text summarization, transformer models

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
BLEU	Bi-Lingual Evaluation Understudy
IBM	International Business Machines
NN	Neural Networks
CNN	Convolutional Neural Networks
AI	Artificial Intelligence
ATS	Abstractive Text Summarization
PEGASUS	Pre-training with Extracted Gap-sentences for Abstractive Summarization
BERT	Bidirectional Encoder Representations from Transformers
DRGN	Deep Recurrent Generative Decoder
IT	Information Technology
RAM	Random Access Memory
DFD	Data Flow Diagram
IDE	Integrated Development Environment
MSE	Mean Square Error
GUI	Graphical User Interface

NLTK	Natural Language Tool Kit
XP	Extreme Programming

LIST OF FIGURES

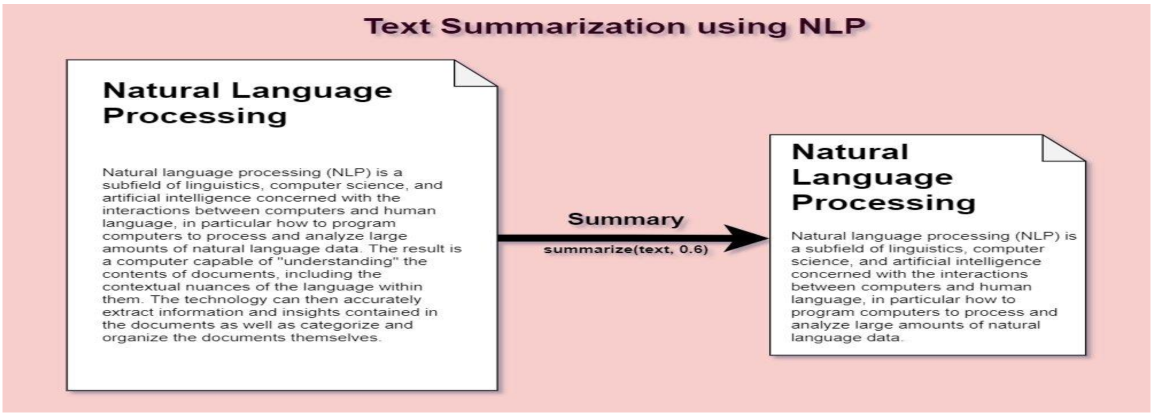
CHAPTER 1: INTRODUCTION

1.1. Introduction

A text that is created from one or more texts that retains the majority of the information in the original text(s) but isn't longer than half of the original text(s) is referred to as a summary. Text summarizing as the process of extracting the key points from a source or sources to create a condensed version that is appropriate for a certain user or users and task or tasks.

Automatic Text Summarization is a difficult subject that is now garnering a lot of attention. When an input is applied, automatic summarization refers to the creation of an output that is automatically summarized. Even though IBM Research Laboratories began researching automatic text summarizing in the 1950s, the Internet has recently caused the area of text summarization to flourish exponentially. Because there is so much information on the Internet, it is exceedingly challenging to manually summarize lengthy texts. The Internet, on the other hand, is a vast library that offers more information than is required. As a result, it is essential for looking through the vast number of documents available for pertinent materials. Text summary seeks to condense the source.

There are two categories of text synthesis techniques. abstract and extroverted summary [1].


Text Summarization using NLP
Natural Language Processing
Summary
summarize(text, 0.6)
Natural Language Processing

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The result is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

Figure 1: Text Summarization using NLP [1]

1.1.1. Abstractive Summarization

When compared to extractive summarization, abstractive summarization is more effective because it gathers data from various publications to produce a precise information summary. Due to the capability of creating new phrases to communicate the crucial information from text documents, this has grown in popularity. A cohesive, easily legible, and grammatically accurate version of the distilled information is presented by an abstractive summarizer. A summary's quality can be raised by boosting its readability or linguistic quality [2].

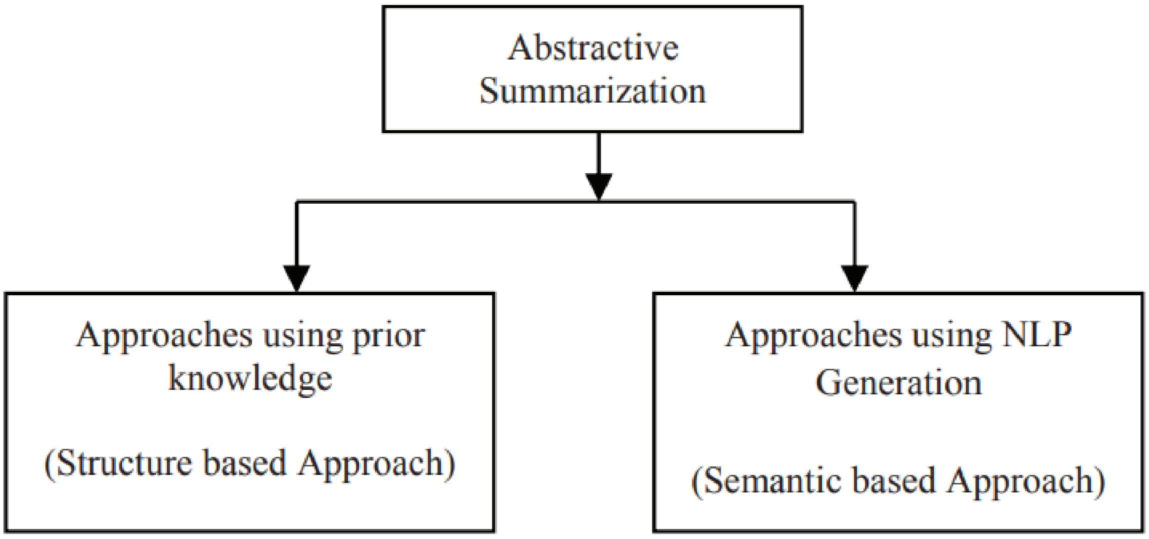

Abstractive Summarization
Approaches using prior knowledge
(Structure based Approach)
Approaches using NLP Generation
(Semantic based Approach)

Figure 2: Overview of Abstractive Summarization [2]

1.1.2. Extractive Summarization

Extractive summarization methods create summaries by selecting a portion of the source text's sentences. The most significant sentences from the input are included in these summaries. A single document or a collection of documents can be the input. In order to provide a better understanding of how summarizing systems function, we present three mostly distinct activities that all summarizers perform. 1) Produce an intermediate text version of the source material that highlights the main concepts. 2) Rate the sentences in light of the example. 3) Select a summary with numerous sentences [3].

1.2. Problem Statement

The problem of abstractive text summarization is to generate a shorter version of a given text document that preserves the most important information and meaning of the original text. This is done by creating new phrases and sentences that are not present in the original text, as opposed to extractive summarization which only selects and combines parts of the original text. Abstractive summarization is a challenging task that requires a deep understanding of the text and the ability to generate a coherent and fluent summary. This task is often tackled with deep learning techniques such as recurrent neural networks (RNNs) and transformer-based models.

1.3. Objectives

This project focuses on condensing a large amount of text into a shorter version while maintaining the main ideas and concepts and generating a summary that is coherent and fluent in terms of language. The primary objectives of our project are listed below:

- To generate summaries that are concise, readable, and grammatically correct.
- To compare the different text summarization models.
- To only take the text's most important information.

1.4. Project Scope and Limitations

Scope and Limitation of Abstractive Text Summarization Abstractive Text Summarization is a powerful Natural Language Processing method that can summarize blocks of text into a condensed version. However, it is not without its limitations.

Scope:

- Abstractive Text Summarization can quickly and accurately generate summaries from large blocks of text.
- It can identify key topics and themes in the text and generate summaries that reflect the overall message of the original text.
- It can also identify similar phrases and sentences and combine them into a single, concise summary.

Limitations:

- Abstractive Text Summarization is a complex process and is not always accurate.
- It is not able to capture all the nuances and details of the original text.
- It is not perfect at generating summaries for longer documents, as it can struggle with understanding the context and structure of the text.

1.5. Development Methodology

The system is developed using Google colab for model training. Python is chosen as a Programming language and documentation is the primary concern. The model will be then deployed into a small web application that will help to execute all the tasks. The summary of the language and libraries used are as follows:

The proposed system is a web application that allows the summarization of text, conversation, and news articles. The proposed system is a web application that makes use of artificial intelligence that makes it cost-effective and easy to maintain. The system makes use of the RNN algorithm for natural language processing and text summarization.

The system is a web application that would summarize the text, conversation, and news article, and the design of the system is divided into three phases as follows:

1.5.1. Model Building

The first phase includes a series of steps of building, training, and testing the model. With the help of the NLTK library, NLP was carried out using various text-processing libraries for classification and tokenization. The datasets were taken from the Hugging face website both for training as well as testing.

1.5.2. Web Application Development

The web application is built to provide a user-friendly interface with the help of HTML/CSS, JavaScript, and Bootstrap for the frontend. The design of the landing page where users are asked to provide the text, article, and conversation to be summarized and the output page were made via the same. The application is developed with the aim that anyone could interact with it.

1.5.3 Model Integration

After the successful development of the model and the web application, the model is integrated into the web and deployed.

1.6. Report Organization

The first chapter includes Introduction, Problem Statement, Objectives, Scope, Development Methodology, and Report Organization. The second chapter consists of a Background Study and Literature Review of the project as well as existing similar projects and research. Chapter 3 includes the analysis part such as Requirements Analysis and Feasibility Analysis (Technical, Economic, Operational, etc.). Chapter 4 includes the design of the system such as ER designs. The fifth chapter is about the actual implementation of the system and its test cases. The final chapter includes the conclusion and future recommendations if any.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

2.1.1. Neural Network

A Neural Network (NN) is a machine learning algorithm that is modeled after the functioningof the human brain. It is composed of neurons, or nodes, that are connected and interact with each other to produce a desired output. The nodes in a neural network are typically organizedinto layers and the connections between nodes are weighted. The strength of the weights determines the output of the network. Numerous tasks, such as image identification, language processing, and forecasting, can be accomplished using neural networks [4].

2.1.2. RNN

Recurrent neural networks (RNNs) are a broad category of dynamic models that have been applied to produce sequences in a variety of fields, including music, text, and motion capture data. RNNs can be trained for sequence production by processing real data sequences one step at a time and predicting what will happen next. Novel sequences can be created from a trained network, providing the predictions are probabilistic, by iteratively sampling from the output distribution and then feeding the sample as input at the subsequent phase. By telling the network to treat its inventions as if they were real, in other words, like a person dreaming. The distribution over sequences is caused by the randomness supplied by selecting samples, despite the fact that the network is deterministic in and of itself. This distribution is conditional since the internal state of the network, and therefore its predictive distribution, depends on the earlier inputs [5].

2.1.3. Deep Learning

Deep learning is a branch of artificial intelligence (AI) that utilizes algorithms to learn from massive amounts of data in a way that is inspired by the structure and operation of the brain. Deep learning modelsare trained by using large amounts of labeled data and neural network architectures that learnfeatures directly from the data without feature engineering. This allows the models to learn complex tasks such as object recognition, natural language processing, and speech recognitionfrom data [6].

2.1.4. NLP

In order to accomplish human-like language processing for a variety of activities or applications, natural language processing (NLP) is a theoretically motivated set of computational approaches for evaluating and representing naturally occurring texts at one or more levels of linguistic analysis. NLP aims to "accomplish human-like language processing," as stated above. The word "processing" was purposefully chosen; it should not be changed to "understanding." A complete NLP system would be capable of:

1. Rephrasing an input text;
2. Translation into another language.
3. Respond to inquiries concerning the book's content;
4. Extrapolate conclusions from the text [7].

2.1.5. Encoder-decoder Framework (Sequence-to-sequence model)

Sequence-to-sequence can be thought of as the first and most fundamental Abstractive Text Summarization (ATS) model. Many of the ATS models that have been presented have improved upon this model. The majority of the architecture of the sequence-to-sequence model is composed of the encoder and decoder, each of which is a Recurrent Neural Network (RNN), as seen in the image. One word at a time is sent into the encoder as input. Each word is first processed by an embedding layer, which turns it into a distributed representation. The preceding word or all 0s for the first word in the text is then given to the hidden layers of a multi-layer neural network in order to integrate this scattered representation. During the decoding process, the hidden layers that are created after feeding in the last word of the input text are used as input. Once more, a distributed representation is created from an input end-of-sequence symbol using an embedding layer. Following that, the decoder produces the text summaries [8].

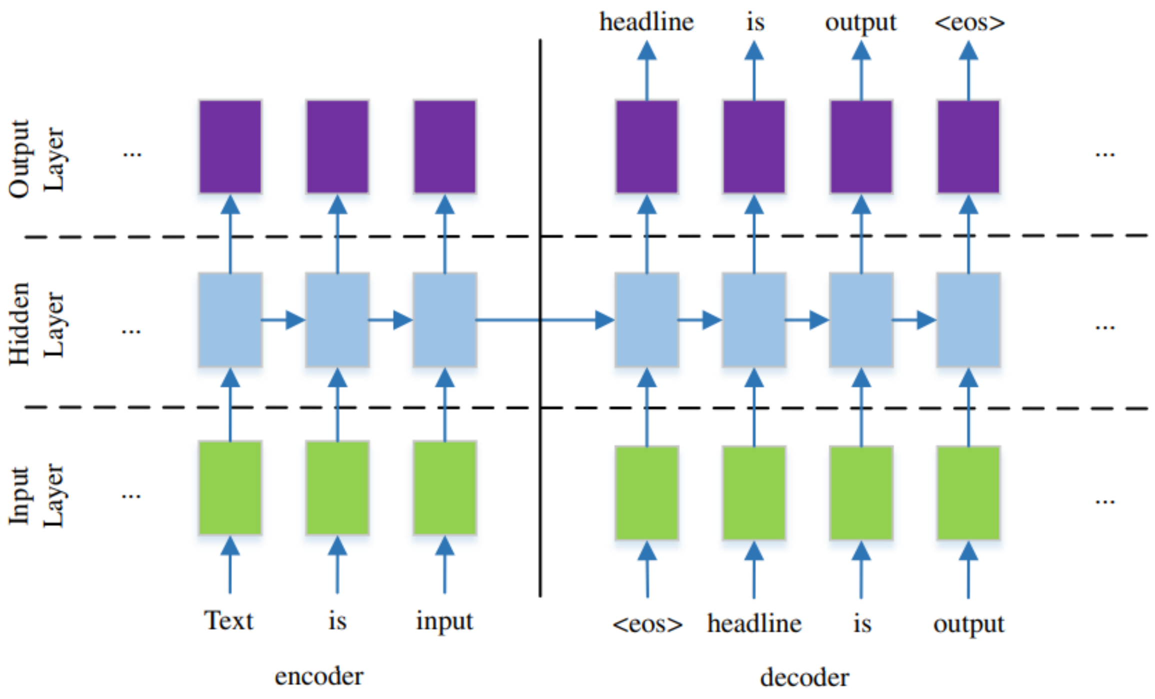


Figure 3: Sequence-to-sequence model [8]

2.1.6. Auxiliary Attention

The majority of existing encoders first read and understand the source text before producing the summary. They are unable to authenticate the accuracy of encoded information, however, or outline key information. Humans, on the other hand, are able to read, comprehend, and highlight text in order to create summaries of the key points. In other words, while producing a summary, people prefer to concentrate on the key details rather than read the entire source text. The existing attention methods, which are normally used in the decoder along with the summary creation and are modeled after how human brains generate summary, are to be replaced with auxiliary attention in the encoder [9].

2.1.7. Transformers

Transformers are a type of neural network architecture that have been used in natural language processing, including abstractive text summarization. They are designed to process entire sequences of text at once, making them more efficient and accurate. The attention mechanism, which is at the heart of the transformer design, enables the model to evaluate the significance of various words in a sequence based on how they relate to other wordsUsing the attention mechanism, the model creates a summary of the text throughout the decoding process that highlights its main ideas. The model is taught to reduce the difference between a human-written summary and one it created [10].

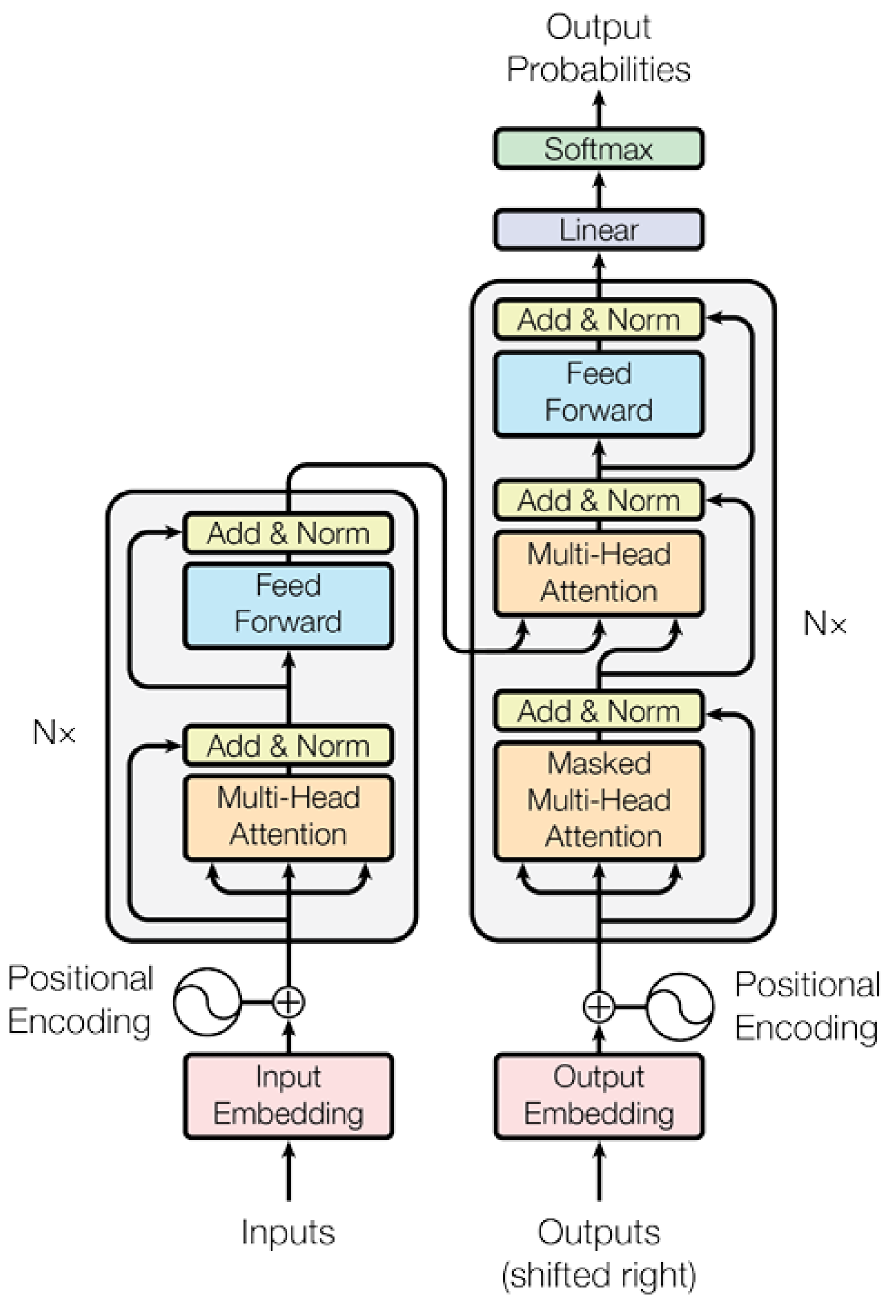


Figure 4: The Transformer - model Architecture [10]

2.2. Literature Review

An ATS framework (ATSDL) based on LSTM-CNN may produce concise summaries by merging information from source sentences and compressing it into a shorter representation while keeping information content and overall meaning. The ATSDL has two steps: the first creates text summaries using deep learning, and the second extracts phrases from source sentences. The framework outperforms state-of-the-art models in terms of semantics and grammatical structure, according to evaluations conducted on the CNN and DailyMail datasets. Additionally, it has generated results that are competitive when utilizing a manual method for linguistic quality assessment [8].

The volume of textual data has been shown to explode recently, necessitating the use of artificial text summarizing models that can take a chunk of text and reduce it to a shorter version. The development of unsupervised and machine learning-based methods for text summarization has advanced significantly with the introduction of deep learning. However, models for text summarization based on deep learning are still in their infancy and have not yet undergone a thorough investigation. A unique abstractive summarization model uses convolutional neural networks, long short-term memory, and supplementary attention in the encoder to improve saliency and coherency. The CNN Daily Mail and DUC-2004 datasets were used to evaluate a model, and the results showed that it performed better in terms of ROUGE metrics, as well as having higher saliency and readability, according to human evaluation [9].

Neural network models based on the attentional encoder-decoder model with a good ability tosummarize the text abstractly have been used. However, they are hard to manage while generating, resulting in a lack of essential data. Hence, a guiding generation model that blendsextractive and abstractive techniques were introduced. Firstly, important words were generatedfrom the text using an extractive model and a Key Information Guide Network (KIGN) was applied to code the words into a significant information representation to guide the generation.Moreover, the prediction-guide mechanism was used to give the long-term value needed for decoding to further guide the summary creation. The model was tested on the CNN/Daily Maildataset and the results show a significant improvement [11].

For neural abstractive summarization, a growing number of people are currently using sequence-to-sequence models. There are many techniques that have been used to raise the fluency, readability, and relevance of seq2seq models. Network structure, parameter inference, and decoding/generation are three areas into which these techniques can be divided. Additionally, techniques to improve the effectiveness of model training and similar are also investigated. This paper gives a broad overview of seq2seq models for abstractive summarizing in terms of network structure, training methods, and algorithms for producing summaries. It also examines how the initial models for tasks like machine translation and language modeling and generation are used to abstractive summarization. Additionally, a free library created expressly for testing called the NATS toolkit [12].

A neural network and continuous sentence attributes are used in a data-driven method for extractive summarization. This technology enables users to build summarization models to extract phrases or words. It comprises a hierarchical document encoder and an attention-based extractor. After being trained on vast datasets, including thousands of document-summary pairings, the models achieved the highest criteria without any linguistic annotation [13].

This study develops and evaluates a novel self-supervised strategy for pre-training Transformer-based encoder-decoder models utilizing huge text corpora. This method, known as PEGASUS, works similarly to an extractive summary by first extracting, then regenerating, essential sentences from an input material. In 12 downstream summarization tasks across several fields, the model was tested, and it consistently received the highest ROUGE ratings. Furthermore, it outperformed the most recent state-of-the-art results on 6 datasets with just 1000 samples. According to the model's summaries' assessments by real humans, the authors' claims were accurate to a human level [14].

According to a study, neural abstractive summarization can be used to create abstractive summaries of lengthy materials with more than a thousand words. It started with the execution of a straightforward extractive phase, and the outcome was applied to condition a transformer language model. The method achieved better ROUGE scores while producing more abstractive summaries than earlier work using a copy technique. The provided model was evaluated against robust baseline techniques, cutting-edge research, and numerous iterations of the methodology. The model was evaluated using four distinct summarizing tasks and datasets, including papers from arXiv, papers from PubMed, papers from the Newsroom, and datasets from BigPatent. According to the results, transformer-based techniques produced summaries with fewer n-gram copies, putting the statistics for n-gram duplication closer to those of abstracts created by humans. Transformers receive excellent marks from the public for coherence and fluency, whereas simply extractive approaches receive greater marks for informativeness and relevance [15].

It was suggested to use neural abstractive summarization to produce abstractive summaries of lengthy materials with word counts up to several thousand. This method begins with a simple extraction step and then conditions a transformer language model with the output. The outcomes of this strategy are more abstract than those of earlier research that employed a copy process, yet they nevertheless get higher ROUGE ratings. The models were assessed by being contrasted with robust baseline approaches, the most recent work, and different iterations of the strategy. The evaluations made use of four distinct summary tasks and datasets, including databases from the arXiv, PubMed, Newsroom, and BigPatent. The findings shown that transformer-based methods produce summaries with fewer n-gram copies, making the statistics of n-gram copying more comparable to those of summaries created by humans [16].

A type of Recurrent Neural Network (RNN) was designed to extractive summarize documents.The model outperformed existing state-of-the-art models and has the added benefit of being transparent. It can be visualized according to information content, salience, and novelty. Furthermore, it is the first extractive model which can be trained in an abstractive fashion, using only human-created reference summaries, and does not require sentence-level extractivelabels [17].

A BERT-based document-level encoder that captured the document's semantics and generated representations for each sentence was demonstrated. A number of intersentence Transformer layers are added to the encoder before the extractive model is constructed. In order to deal with the problem of the encoder being pre-trained while the decoder is not, a novel fine-tuning schedule for abstractive summarization was published. For the encoder and decoder, different optimizers were utilized. It offers a two-step process for fine-tuning that can improve the caliber of the produced summaries. On three datasets, the model produced cutting-edge findings in both extractive and abstractive situations [18].

As a new framework for abstractive text summarization, a deep recurrent generative decoder (DRGN)-equipped, sequence-to-sequence oriented encoder-decoder model has been created. Latent structure information inferred from the target summaries is trained to improve the quality of the summarizing based on a recurrent latent random model. The unsolvable posterior inference for the recurrent latent variables is addressed using neural variational inference. On the basis of both the discriminative deterministic states and the generative latent variables, abstractive summaries are produced. Extensive tests on benchmark datasets in several languages demonstrate that DRGN outperforms state-of-the-art techniques [19].

In the sequence-to-sequence framework, the difficulties of neural abstractive document summarization were examined along with a novel graph-based attention mechanism. The goal is to address the saliency part of summarization, which has been overlooked in past attempts. Experimental findings show that the model can significantly outperform earlier neural abstractive models. Additionally, the data-driven neural abstractive method competes favorably with cutting-edge extractive techniques [20].

For the abstractive sentence summarization problem, which tries to provide a summary of a given source sentence, a contrastive attention mechanism to extend the sequence-to-sequence framework was presented. Two types of attention are accommodated by the suggested contrastive attention mechanism: the opponent attention, which pays attention to less important or irrelevant sections of the source sentence, and the conventional attention, which pays attention to relevant elements of the source sentence. Through the use of innovative softmax and softmin functionalities, both attentions are taught in opposition to each other, encouraging the input of the conventional attention and discouraging the participation of the opponent attention. The proposed contrastive attention mechanism is more focused on the key components for the summary than the usual method, according to experiments on benchmark datasets [21].

LTABS (Abstractive summarization on LSTM and Transformer), a novel seq-to-seq text summarization model, was maintained. The transformer is more suited to tasks requiring summary generation thanks to this model. The hidden layer output of the LSTM was used as the position information for the encoder portion of the model. The transformer doubles as the LSTM's attention matrix at the same time. The transformer can learn its location thanks to its design, which also improves local awareness. To solve the OOV (out of vocabulary) problem in summary tasks, a copy mechanism to the new network was added. The test results indicate that the LTABS framework is superior to the most advanced model in the datasets used for CNN, Daily Mail, and Xsum [22].

For the purpose of abstractive text summarization, a generative model G and a discriminative model D were simultaneously trained in an adversarial method. In particular, the reinforcement learning agent generator G was created, which forecasts abstractive summaries using raw text as its input. A discriminator that seeks to distinguish between the generated summary and the ground truth summary was also built. Numerous experiments demonstrate that the model provides competitive ROUGE scores using state-of-the-art methods on the CNN/Daily Mail dataset. The model was able to create summaries that were qualitatively more abstractive, readable, and diverse [23].

A dual encoding paradigm was used to simulate an abstractive text summary. The suggested approach uses a dual encoder, including the primary and secondary encoders, in contrast to earlier studies that simply used a single encoder. While the primary encoder frequently performs coarse encoding, the secondary encoder models the importance of words and offers more fine encoding depending on the input raw text and the previously generated output text summary. To provide a more diverse summary that can lessen the repeating phenomenon for the development of prolonged sequences, the two-level encodings are concatenated and fed into the decoder. The experimental results on the challenging datasets CNN/DailyMail and DUC 2004 demonstrate the effectiveness of the proposed dual encoding model in comparison to other approaches [24].

It was suggested to use a model-based, weakly supervised method to check for consistency in the facts and spot inconsistencies between the source texts and a derived summary model. To create training data, a set of rule-based transformations are applied to the source documents' sentences. After that, three tasks are jointly trained for the factual consistency model: Once the phrases have undergone transformation, check to see if they are factually consistent, and if so, extract a span from the original papers to back up your conclusion. If there is an inconsistent span in the summary sentence, then extract it. When used to analyze summaries generated by various state-of-the-art techniques, this extremely scalable method greatly outperforms prior models, even those trained under close supervision using standard datasets for natural language inference and fact-checking. The human evaluation also demonstrates that the auxiliary span extraction tasks are helpful in the process of confirming factual consistency [25].

By addressing the various ATS components—approaches, methods, building blocks, procedures, datasets, assessment techniques, and future research directions—the study offers a thorough review for researchers. Due to the enormous amount of textual content, manual text summarizing requires a lot of time, effort, and money and can even become unfeasible. Since the 1950s, scientists have been working to develop ATS procedures. Extractive, abstractive, and hybrid ATS approaches are the three different categories. The extractive technique chooses the key terms from the source document(s) and concatenates them to create the summary. The abstractive approach represents the input document(s) in an intermediate form before creating the summary using sentences that are different from the source language. The hybrid approach combines extractive and abstractive techniques [26].

To formally model and enhance the information selection process in abstractive document summarization, it was proposed to add an information selection layer to the basic neural encoding-decoding architecture. In particular, the information selection layer is divided into two sections: local phrase selection and gated global information filtering. The unnecessary portions of the original text are first globally filtered; then, the most crucial sentences are selected locally, and finally, each summary phrase is created one at a time. Additionally imported is distantly-supervised training directed by the golden summary in order to directly optimize the information selection method. Due to the explicit modeling and optimization of the information selection process, which greatly surpasses state-of-the-art neural abstractive approaches, the model may provide summaries that are significantly more informative and concise. Experimental results support this claim [27].

CHAPTER 3: SYSTEM ANALYSIS

3.1. System Analysis

Systems analysis is the method through which a person or group of people examines a system in order to model, evaluate, and choose a logical replacement for an information system. The initiating factors for systems analysis activities are three: issues, opportunities, and directives. Users, sponsors, and system analysts are all involved. The life cycle of systems development can be used to explain the creation of systems. A methodology is a term that can be used to describe the tasks, procedures, and equipment employed within the systems development life cycle. Methodologies can be divided into three categories: conventional, information engineering, and object-oriented. CASE tools support particular approaches and are automated tools. It is performed by utilities to plan and develop electric power networks. New equipment can be simulated using system analysis methods, reducing uncertainty about its performance in the particular application for which its installation is planned. A thorough examination of the existing system will be carried out to determine the feasibility of designing a new system that addresses identified problems, satisfies specified requirements, and improves upon any shortcomings of the current system [28].

3.1.1. Requirement Analysis

When it comes to software projects, requirements analysis is the process of researching, identifying, and recording user needs and expectations for the software system that will be developed to address a certain issue. It includes feasibility studies, elicitation, specification, and validation process steps. The process is known as requirements engineering. The procedure creates software requirement documents that thoroughly describe the functionality, performance, design limitations, and quality aspects of the software system in order to capture what must be implemented. Clarity in the documentation of what to develop helps to lessen ambiguity and uncertainty. Undoubtedly, the secret to creating successful information systems is figuring out and capturing the needs. The expense and timeline of the project are largely attributable to the wrong final software system requirements being obtained at the beginning of the project [29].

3.1.1.1. System Requirements

The system necessitates a specific set of hardware and software components to be utilized. These requirements and specifications are listed as follows:

Hardware Requirement:

- RAM: 8 GB (minimum)
- Processor: Intel Core i3 or higher
- Input device: Keyboard, Mouse
- Output device: Monitor

Software Requirements:

- Code Editor: VS Code
- Web Browser: Google Chrome
- IDE: Pycharm/GoogleColab
- Draw.io
- MS Word
- Google Drive
- OS: Windows 8 or newer version

3.1.1.2. Functional Requirement

The desired operations of a program or system are referred to as functional requirements in software development and systems engineering. A combination of software-driven electronics and electronic hardware can be used as the systems in systems engineering. Requirements engineering, often known as requirements analysis, is a multidisciplinary area of engineering that deals with the development and maintenance of complex systems. The examination of requirements includes functional requirements. Functional requirements define the anticipated end function of a system functioning within normal parameters to ensure the design is adequate to provide the desired outcome, the final product achieves its potential, and the design to satisfy user expectations [30].

Functional requirement of our system:

User-Interface: an interface was developed where the user was given an option based on inputwhich redirected to the output page.

Training data: The existing model was used to train different datasets to generate summariesfor various types of inputs.

Ability to handle different types of input: The model should be able to handle different types of input text, such as news articles, scientific papers, or conversations.

Use Case Diagram

A use case diagram illustrates how a system behaves dynamically. To encapsulate the system's functionality, it includes use cases, actors, and their interactions. It reproduces the responsibilities, functions, and actions required by a system or application subsystem. It demonstrates the basic operations of a system and defines user interactions. The main objective of a use case diagram is to demonstrate the dynamic nature of a system. The requirements of the system are accumulated, taking into consideration both internal and external elements. It displays how an object from the outside world might engage with a component of the system.

The use case diagram's objectives are listed below:

- It depicts the system's external perspective and collects the system's demands.
- It acknowledges both internal and external influences on the system.
- It depicts how the performers interacted with one another [31].

The use case diagram illustrates the functional requirements of the system, The user input text through the interface. The admin handles the labeled data and trains the model using the dataset. The model then preprocesses the input text, tokenizes the input, and finally generates the appropriate summary. The system finally displayed the output which is presented to the user.

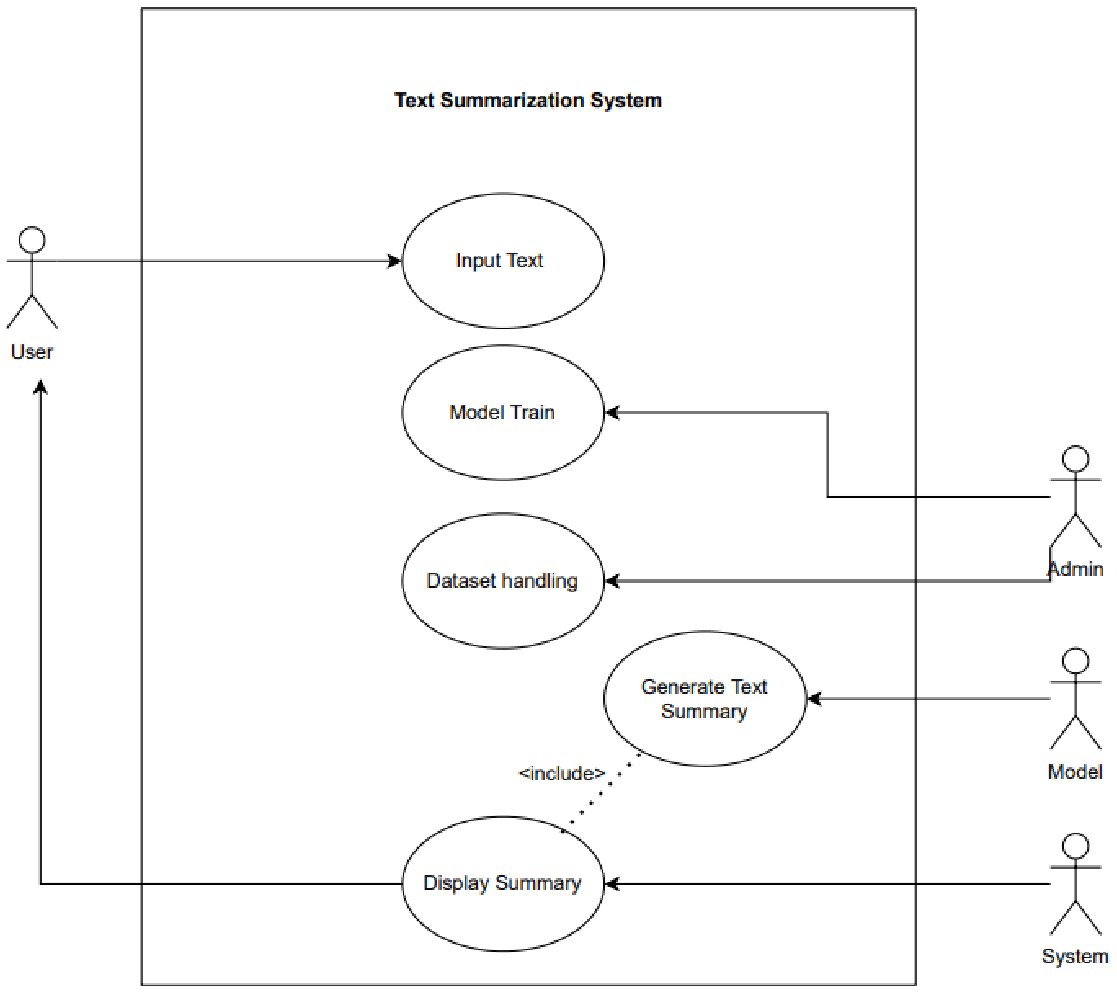


Figure 5: Use Case Diagram

3.1.1.3 Non- Functional Requirements:

The IT system's non-functional requirements are those elements that, while not directly affecting the application's business functionality, have a significant impact on end users' and program support staff's perceptions of the efficiency and effectiveness of business systems. An key component of creating a comprehensive customer solution that achieves organizational objectives is the specification of these needs. In order to ensure the durability and toughness of the application, the non-functional requirements (NFRs) are largely employed to drive the operational aspects of the architecture [32].

Usability: The software must be straightforward and simple to use. The program must display all pertinent facts and relationships in an understandable and straight forward manner.

Reliability: When there is a crash or error, the program must alert the user. Should be willing to work as and when needed.

Performance and scalability: The system runs pretty fast and can return the result of any text at a quick speed. The speed heavily relies on the type of system used. Higher the specifications of the system faster the program runs.

Portability and compatibility: The system can run on a system with RAM of 8 GB or higher and a 1 GHz or faster processor. These are readily available on most systems nowadays. During development, it runs on the Windows platform but can be further developed to run on Mac and Linux. Python is available for all platforms. All of the tools are available for cross-platform portability and will have no issues with compatibility.

3.1.2. Feasibility Analysis

A feasibility study offers the chance to "firm up" one's understanding of the system and to develop concepts on the breadth and price of potential solutions. In many ways, the feasibility study is a short and dirty version of a systems analysis, with the analyst focusing on many of the problems and employing many of the methods needed for more in-depth work. Typically, feasibility studies are completed in a short amount of time, and they end with a written and verbal feasibility report. The findings and suggestions of such a research will serve as the foundation for choosing whether to move forward with the project, put it on hold, or abandon it. Because the feasibility study could result in the deployment of significant resources, it is crucial that it be carried out correctly [33].

A feasibility study is conducted to examine whether the idea is viable in various contexts. This research will help us determine whether the project is feasible and will also assist us in governing various instances where implementation issues may develop.

The study will ensure that no issues develop during the project's real implementation. The results of the feasibility study conducted for this project are detailed below.

3.1.2.1. Technical feasibility

The application is technically feasible as it complies with current technology including both the hardware and software. The web application will be supported by almost all the latest computers with minimum hardware and software requirements.

3.1.2.2. Economic feasibility

The application is economically feasible as it will only require a good internet connection and a computing device with minimum hardware and software requirements to operate.

3.1.2.3. Operational feasibility

The system will be able to easily summarize the texts, conversations, and articles. This can be done efficiently without needing much expert knowledge making it suitable for people of any background.

3.1.2.4. Legal feasibility

The system abides by cyber ethics and does not cross any legal and ethical boundaries. Data privacy, copyright, and intellectual property are well preserved.

3.1.2.5. Schedule feasibility

The diagram below shows the project timeline for the proposed system. The timeline includes a series of steps from project initiation to final documentation. Each step in the timeline is supposed to be carried out within the given timeframe. With every step conducted strictly to deadlines and moving forward to the next milestone, the system meets the time feasibility constraints.

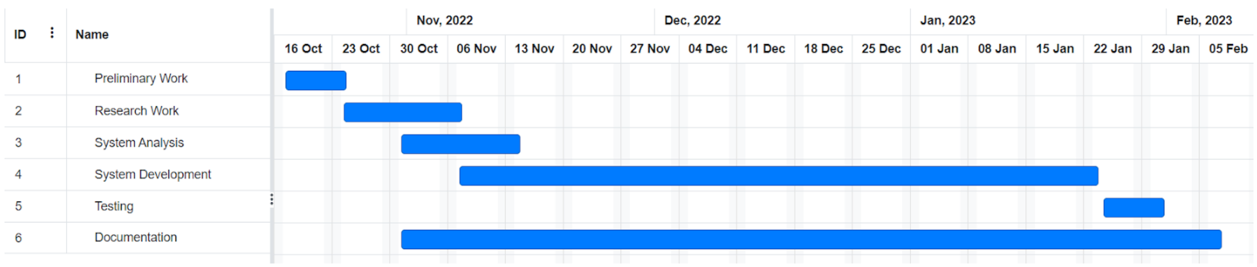


Figure 6: Gantt Chart

3.1.3. Analysis

Data Flow Diagram:

The data flow diagram (DFD) is a technique for structured analysis and design. It serves as a visual tool for representing logic models and data transformation in a system. DFD includes a mechanism for simulating the data flow. To display the characteristics of the data flows and functions, decomposition is supported. DFD cannot provide information on the order of the operations. It cannot be used to represent processes or procedures as a result. The following traits are present in DFD:

(1) Supporting the requirements and analysis phases of system design;

(2) a method for diagramming with annotations

(3) characterizing the target system's network of processes and activities;

(4) allowing for concurrent and asynchronous actions. step-by-step process improvement through process hierarchy decomposition [34].

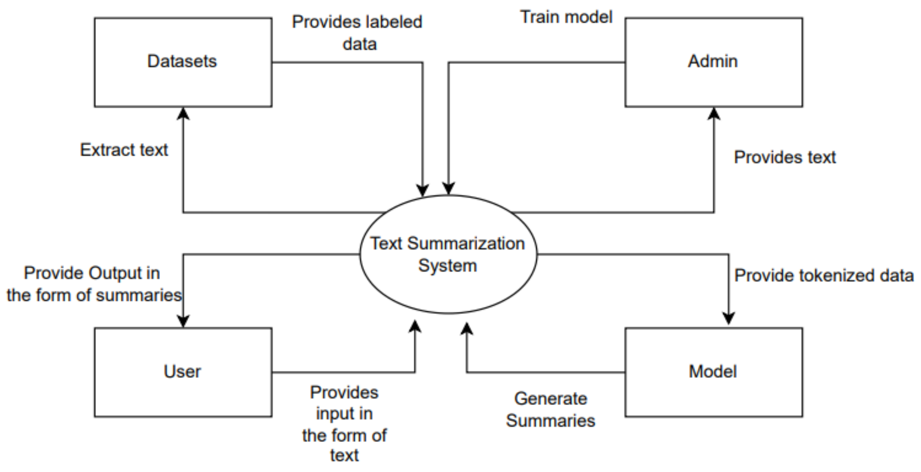


Figure 7: Context Flow Diagram

The user provides input in the form of text through the interface and the text summarization system generates the appropriate summary. The dataset provides labeled data which is used by the admin to train the model. The model then uses the tokenized inputs to generate the appropriate summary.

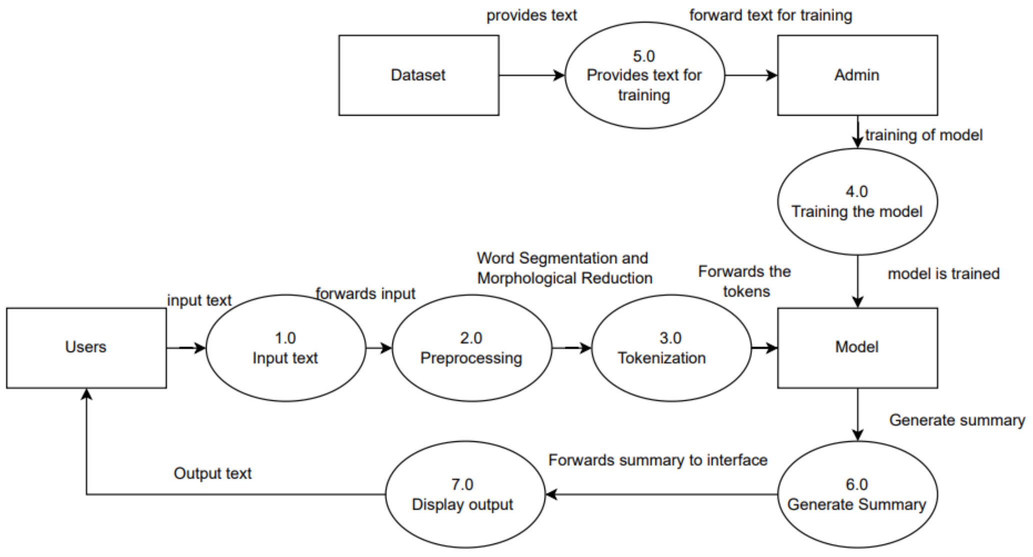
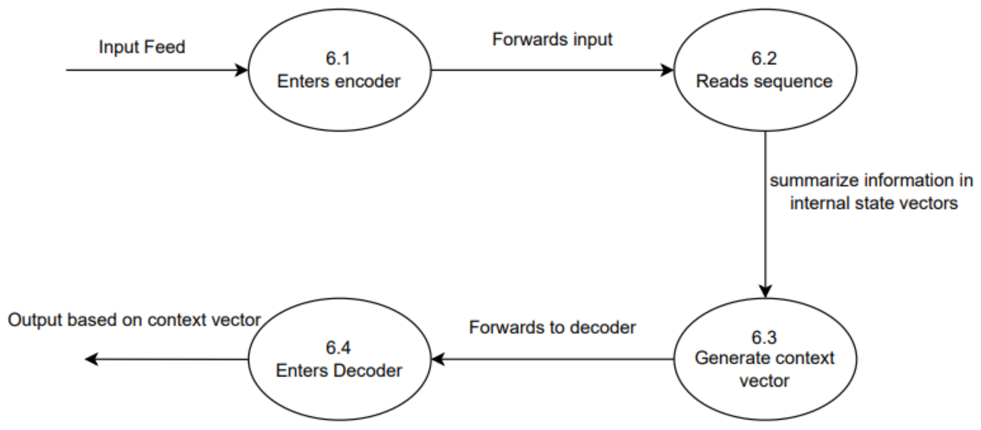


Figure 8: DFD level 1

The user inputs the data from the interface. The input data is preprocessed using Word Segmentation and Morphological Reduction. The preprocessed data is then tokenized which is used by the model. To train the model, the dataset is used to provide labeled data and the admin is used to then train the model using that data. The model then uses


Input Feed
6.1 Enters encoder
Forwards input
6.2 Reads sequence
summarize information in internal state vectors
6.3 Generate context vector
Forwards to decoder
6.4 Enters Decoder
Output based on context vector

the tokenized input to generate the summary. The summary is then forwarded to the interface. Finally, the output is displayed to the user via the interface.

Figure 9: DFD level 2

The input is fed into the encoder. Input sequence is read by the encoder, which then creates internal state vectors that are utilized to create context vectors. The decoder receives the context vector after that. Finally, decoder produces output using the context vectors.

CHAPTER 4: SYSTEM DESIGN

4.1. Design

4.1.1. System Flowchart

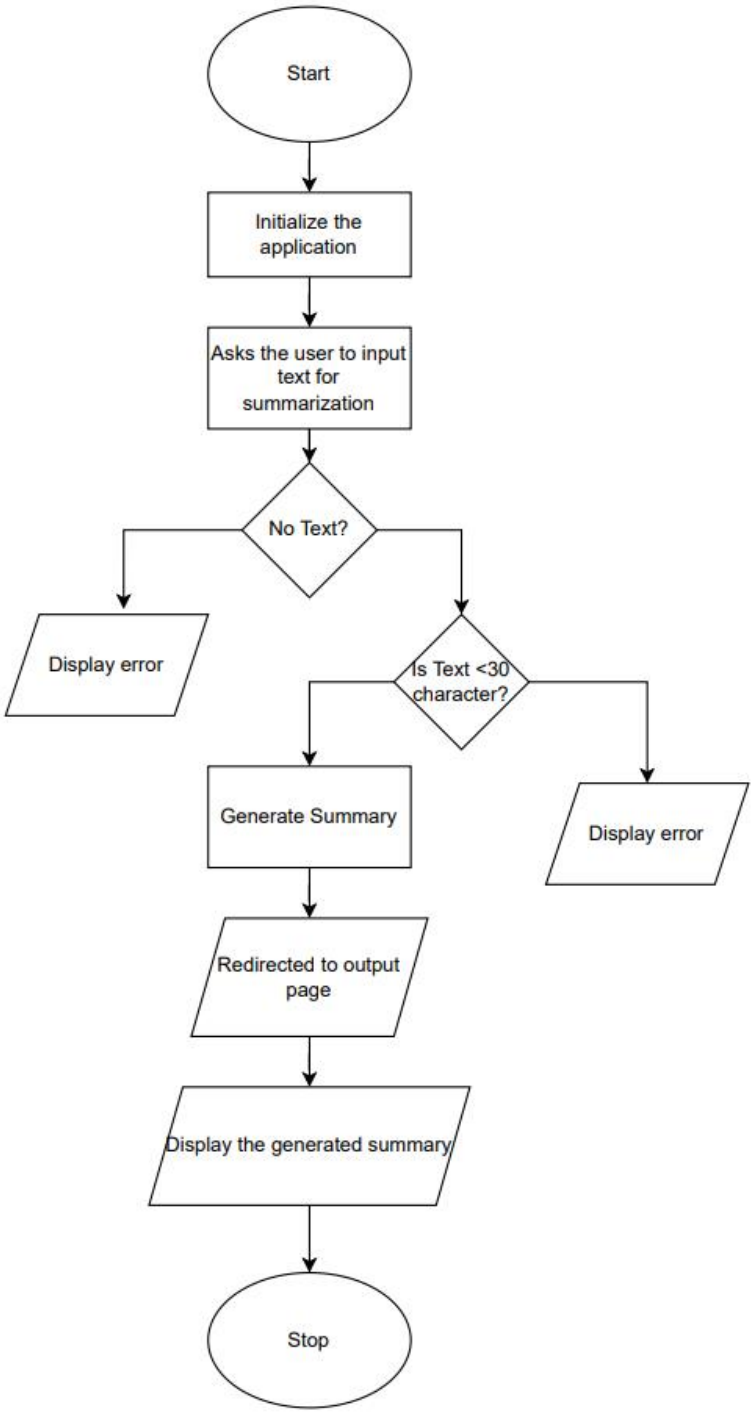

Start
Initialize the application
Asks the user to input text for summarization
No Text?
Display error
Is Text <30 character?
Generate Summary
Display error
Redirected to output page
Display the generated summary
Stop

Figure 10: System Flowchart

The application is initialized at the start. The user is asked to input the text for summarization. If no text is inputted, then an error message is displayed. Similarly, if an input consists of only 30 characters, then an error message is displayed. Otherwise, the summary is generated by the system. The output is shown after it is redirected to another page. Finally, the output is displayed on the screen. Then the system is stopped.

4.1.2. Form and Report Design

The application is an interface that contains a textbox for input and a button "Text Summarization" which generates the required summary.

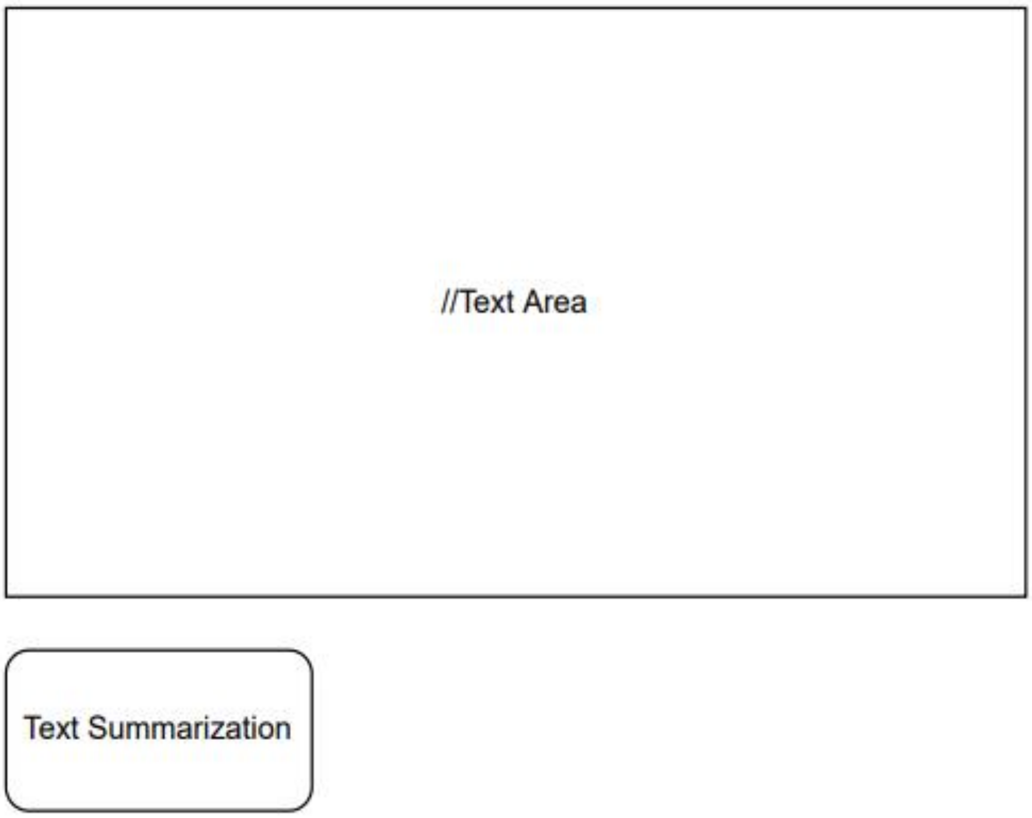
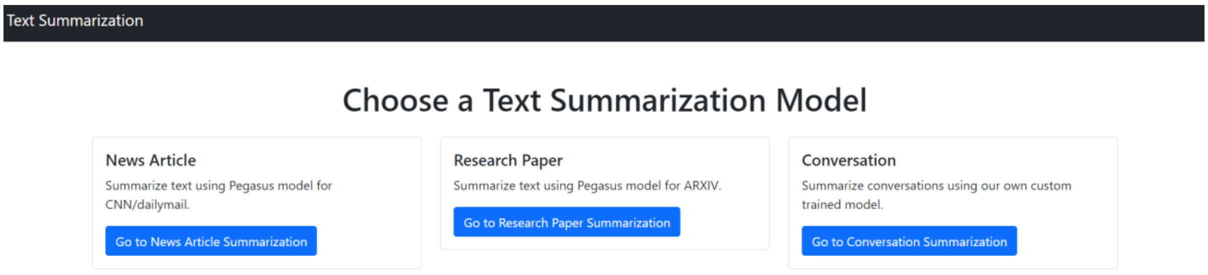


Figure 11: Form Design

The output page is redirected from the main page which displays the generated summary.

4.1.3. Interface Design


Text Summarization

Choose a Text Summarization Model

News Article

Summarize text using Pegasus model for CNN/dailymail.

Go to News Article Summarization

Research Paper

Summarize text using Pegasus model for ARXIV.

Go to Research Paper Summarization

Conversation

Summarize conversations using our own custom trained model.

Go to Conversation Summarization

Figure 12: Interface Design

The final interface can be seen in the figure. The menu can be seen with the Text Summarization as the heading. Then a textbox is seen where the text is inputted. Finally, to generate a summary, the button named ‘Text Summarization’ can be used. The button redirects to the output page where the summary is displayed.

4.2. Algorithm Details

4.2.1. Attention algorithm

Humans require the complicated cognitive ability of attention, which is essential. Humans do not typically process all of the information they are presented with at once, which is a crucial characteristic of perception. Instead, when and when it is necessary, humans prefer to selectively focus on a portion of the information while ignoring other perceptible information at the same time. Human attention mechanisms can be split into two groups based on how they are generated. The first group is saliency-based attention, or bottom-up unconscious attention, which is motivated by extraneous stimuli. Top-down conscious attention, also known as focused attention, falls under the second type. The word "focused attention" describes focus that is directed toward and dependent upon particular tasks. It helps people to actively and intentionally direct their attention toward a certain item. Deep learning's attention mechanisms are mostly focused because they are created with certain objectives in mind. The major method for addressing the issue of information overload is to use the attention mechanism as a resource allocation technique. It can process more significant data with less computing resources if there is a shortage of computing power [35].

There are different models used by this algorithm for text summarization. Sequence-to-sequence and PEGASUS are used in this project.

4.2.1.1. Sequence-to-Sequence

Sequence-to-sequence models, which use deep learning to map an input sequence into an output sequence, have shown promise for solving a variety of issues, including speech recognition, machine translation, and video captioning. The attentional Recurrent Neural Network (RNN) encoder-decoder model, which has generated state-of-the-art performance in machine translation (MT), which is also a natural language job, is a particularly relevant model to the task within the context of sequence-to-sequence models [36].

4.2.1.2. Pegasus

The architecture of the PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization) model is Transformer based encoder-decoder network and the model targets to output the important sentences by masking principal sentences or greedily selected sentences based on the ROUGE in the input text during the training process [37].

4.2.1.3. ROUGE Metric

The acronym ROUGE stands for Recall-Oriented Understudy of Gisting Evaluation and refers to a collection of criteria for assessing automatically generated texts. It is usually used to evaluate the quality of a TS algorithm's summary. To operate, ROUGE compares a machine-generated summary (sometimes referenced to as a system summary) to one created by a human (sometimes called gold standard or reference summary). The ROUGE metric refers to a set of distinct ways to quantify the quality of a system summary. ROUGE measures can be calculated in a variety of methods, based on the different granularity. The following are the most commonly used: ROUGE-N, ROUGE-L, ROUGE-S, and ROUGE-SU. ROUGE-1, ROUGE-2, and

ROUGE-L are the most commonly used metrics in the literature since they reflect the granularity of the studied texts [38].

4.2.1.4. Scare BLEU Score

A parameterized metric called BLEU (Bi-Lingual Evaluation Understudy) has values that are extremely sensitive to changes in these parameters. The BLEU scores between articles cannot be directly compared because these metrics are frequently not given or are difficult to locate. BLEU has been adopted as the standard measurement for machine translation research due to its relative language independence, simplicity of computation, and reasonable connection with human judgements. Overall, it has benefited the community by giving academics a quick and affordable tool to evaluate the effectiveness of their models. In addition to more extensive controlled manual inspections, BLEU has guided the field through 15 years of quality advancements [39].

4.2.2. Model Building Algorithm

Step 1: Start
Step 2: Import the required libraries such as Transformer, and Torch.
Step 3: Load the dataset for processing
Step 4: The ROUGE score of the dataset was calculated for the model.
Step 5: The necessary fine-tuning was done and the model was trained.
Step 6: The model and the tokenizer were saved.
Step 7: Load the test data in the model and evaluate the trained model with the ROUGE score
Step 8: The necessary summary was generated
Step 9: Stop

4.2.3. System Algorithm

Step 1: Start
Step 2: User launches the application.
Step 3: User is asked to choose one of the three options which depends on the inputted data. The options are news articles, research papers and dialogue between people.
Step 4: If the user chooses one of the options
4.1 A new page is opened with a textbox and button

4.2 The system performs text summarization on the preprocessed data

4.3 A new page is opened which displays the generated output.

Step 5: The output is displayed to the user.

Step 6: Stop

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1. Implementation

5.1.1. Tools Used (CASE tools, Programming languages, Database platforms)

The following hardware and software tools are used to develop the sentiment analysis system:

Hardware tools

- 8 GB RAM or higher.
- 1 GHz or faster processor.
- Input device: Keyboard, Mouse
- Output device: Monitor

Software tools

5.1.1.1. PyCharm

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. It supports Python 2 and Python 3. One can develop an application using Django, Flask, and Pyramid as well, and it supports HTML5, CSS, JS, and XML. PyCharm is a cross-platform IDE that works on Windows, macOS, and Linux [40].

The tool provided an easy-to-use environment for coding, debugging, testing, and running applications. It was used to improve the workflow, from writing code to debugging and deploying. It was used to automate tasks, such as code refactoring, navigation, and search. It helped to speed up the development process and makes the entire development process more efficient.

5.1.1.2. MS Word and Excel

MS Word is a word processor created by Microsoft that is used to create papers, letters, reports, and other types of writing of a professional caliber. It includes sophisticated capabilities that give you the best formatting and editing options for your files and projects. Users use MS Word can write articles and generate papers, resumes, contracts, and other things. Microsoft Excel is also a product of Microsoft used extensively for data analysis. It's a spreadsheet application with several columns and rows,with each intersection of a column and a row being referred to as a "cell." It's used to make text, number, and formula grids that specify calculations. This is one of the most commonly used programs under the Office suite [41].

Microsoft Word has been used to document the requirements, process, and workflow of our project from the beginning to the end. Similarly, Microsoft Excel has been used in creating the Gantt chart and keeping track of the time constraints for our project.

5.1.1.3. Google Drive

Google Drive is a free Google file storage service in the cloud. Launched in 2012 to replace Google Docs, which only had 1GB of storage, it meant a wide increase in capacity for users reaching up to 15GB completely free. In addition, Google Drive incorporates various applications such as a spreadsheet program, a program to create presentations, and a text editing program, the 3 very similar to the Microsoft Office package, in this case assimilating to Microsoft Excel, Microsoft PowerPoint, and Microsoft Word. Its simple design and full of utilities have made Google Drive one of the most used services of Google. Among its main features are File Storage, Document creation, Connectivity, Integrated apps, etc [42].

Models are saved and shared in between using Google Drive. The documents, diagrams, and tables are saved in Drive where a folder was made and shared between the authors. The use of Drive made it easier to work remotely and collaborate.

5.1.1.4. Google colab

Google Colaboratory, or “Colab” as most people call it, is a cloud-based Jupyter notebook environment. It runs in your web browser (you can even run it on your favorite Chromebook) and lets anyone with internet access experiment with machine learning and coding for artificial intelligence. You can write and execute Python code, share your code and edit it simultaneously with other team members, and document everything by combining it into a single notebook with rich text, charts, images, HTML, and LaTeX [43].

Google Colab being a free resource has been of great use in our project to train the model with ease. We have used Google colab to train and test our model by providing the necessary dataset and exported the trained model to use in our web interface.

5.1.1.5. Draw.io

A proprietary tool for drawing charts and diagrams was developed by Seibert Media and is called Draw.io. You have the choice to use the automatic layout tool or create a custom layout using the software. They include a large range of shapes and a number of visual elements to assist you in developing a special diagram or chart. The drag-and-drop feature makes it simple to construct a beautiful diagram or chart. Depending on your needs, Draw.io offers alternatives for storing saved charts in the cloud, on a server, or network storage in a data center [44].

Draw.io has been used in the process of creating flowcharts, context diagrams, data flow diagrams, and use case diagrams.

5.1.1.6. Google Chrome

Google Chrome browser is a free web browser used for accessing the internet and running web-based applications. The Google Chrome browser is based on the open-source Chromium web browser project. Google released Chrome in 2008 and issues several updates a year. Google Chrome is available for Microsoft Windows, Apple macOS, and Linux desktop operating systems (Oses), as well as the Android and iOS mobile operating systems. Google Chrome is the default browser for Google devices including Android phones and Chromebook laptops. Google Chrome is part of Google services, Google’s family of products that includes Gmail, Google Maps, and the Chrome search engine. Many of Google's services are hosted on Chrome, which was one of the first internet browsers to use the cloud to support applications and systems which makes it beneficial. It is a widely used browser with facilities of extensions [45].

Google Chrome was used to surf through the internet to have access to various resources such as research papers, design tools, etc.

5.1.1.7. Grammarly

Grammarly is a grammar checker and proofreading tool that goes above and beyond its technical function. In addition to helping with spelling, punctuation, grammar, and sentence structure, it also provides real-time support to help you increase your vocabulary and improve the clarity, cohesiveness, and fluency of your writing. There is a reason Grammarly has emerged as the go-to writing and proofreading tool, even among journalists, because to its features, simplicity of use, accessibility, and highly rated accuracy. Once it is downloaded and enabled on your device or linked to your browser, its AI engine automatically scans your work and highlights any words, phrases, or sentences that could be improved. The red and yellow underlines that indicate the words or phrases that are either inaccurate or can be improved make it simple to identify areas that want modification (yellow underlines indicate difficulties required additional features available exclusively in Premium and Business accounts) [46].

The tool helped to correct all the grammatical errors that occurred during the documentation process.

5.1.2. Implementation Details of Modules

Comparing Different Models

The PEGASUS model is compared with other various text summarization models such as GPT2, BART, and T5. A sample text is taken from a dataset and summarized using all these types of models. The ROUGE score of the summarized output was calculated.

Comparing different models using the Scare BLEU score

```
bleu_metric.add(prediction = [summaries["pegasus"]], reference =
[dataset['train'][1]['highlights'] ])

results = bleu_metric.compute(smooth_method = 'floor', smooth_value
= 0 )
results['precision'] = [np.round(p , 2) for p in
results['precisions'] ]
```

```
pd.DataFrame.from_dict(results, orient = 'index', columns =
['Value'] )
```

Comparing different models using the ROUGE score

```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]

reference = dataset['train'][1]['highlights']
records = []

for model_name in summaries:
    rouge_metric.add(prediction = summaries[model_name], reference =
reference )
    score = rouge_metric.compute()
    rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in
rouge_names )
    print ('rouge_dict ', rouge_dict )
    records.append(rouge_dict)

pd.DataFrame.from_records(records, index = summaries.keys() )
```

Tokenization

```
def calculate_metric_on_test_ds(dataset, metric, model, tokenizer,
                               batch_size=16, device=device,
                               column_text="article",
                               column_summary="highlights"):
    article_batches =
list(generate_batch_sized_chunks(dataset[column_text], batch_size))
    target_batches =
list(generate_batch_sized_chunks(dataset[column_summary],
batch_size))

    for article_batch, target_batch in tqdm(
        zip(article_batches, target_batches),
total=len(article_batches)):
        inputs = tokenizer(article_batch,
max_length=1024,  truncation=True,
                        padding="max_length", return_tensors="pt")
        summaries =
model.generate(input_ids=inputs["input_ids"].to(device),
                         attention_mask=inputs["attention_mask"].to(
device),
                         length_penalty=0.8, num_beams=8,
max_length=128)

        decoded_summaries = [tokenizer.decode(s,
skip_special_tokens=True,
                                clean_up_tokenization_spaces=True)
               for s in summaries]

        decoded_summaries = [d.replace("", " ") for d in
decoded_summaries]

        metric.add_batch(predictions=decoded_summaries,
references=target_batch)
```

Histogram

```
dialogue_token_len = len([tokenizer.encode(s) for s in
dataset_samsum['train']['dialogue']])

summary_token_len = len([tokenizer.encode(s) for s in
dataset_samsum['train']['summary']])


fig, axes = plt.subplots(1, 2, figsize=(10, 4))
axes[0].hist(dialogue_token_len, bins = 20, color = 'C0', edgecolor
= 'C0' )
axes[0].set_title("Dialogue Token Length")
axes[0].set_xlabel("Length")
axes[0].set_ylabel("Count")

axes[1].hist(summary_token_len, bins = 20, color = 'C0', edgecolor =
'C0' )
axes[1].set_title("Summary Token Length")
axes[1].set_xlabel("Length")
plt.tight_layout()
plt.show()
```

Model Training

```
trainer_args = TrainingArguments(
    output_dir='pegasus-samsum', num_train_epochs=1,
warmup_steps=500,
    per_device_train_batch_size=1, per_device_eval_batch_size=1,
    weight_decay=0.01, logging_steps=10,
    evaluation_strategy='steps', eval_steps=500, save_steps=1e6,
    gradient_accumulation_steps=16
)

trainer = Trainer(model=model_pegasus, args=trainer_args,
                  tokenizer=tokenizer,
data_collator=seq2seq_data_collator,
                  train_dataset=dataset_samsum_pt["train"],
                  eval_dataset=dataset_samsum_pt["validation"])
```

Model

```
model_path = "Users\HP\Desktop\Text-summarization-flask-huggingface-
main"
model_name_or_path = "transformersbook/pegasus-samsum"
device = "cuda" if torch.cuda.is_available() else "cpu"
tokenizer = PegasusTokenizer.from_pretrained(model_name_or_path,
cache_dir=model_path)
model =
PegasusForConditionalGeneration.from_pretrained(model_name_or_path,
cache_dir=model_path).to(device)
```

JS for validation

```
<script>
       // get a reference to the form
       const form = document.getElementById('summarization-form');

       // add an event listener for form submission
       form.addEventListener('submit', function(event) {
           // prevent the form from submitting if there are errors
           event.preventDefault();
           // get a reference to the input field
           const inputField = document.getElementById('inputtext_');

           // get the input value
           const inputValue = inputField.value;

           // check if the input value is empty or only contains
whitespace
           if (!inputValue.trim()) {
   // if it's empty or only contains whitespace, display another
error message
   inputField.classList.add('is-invalid');
   inputField.setCustomValidity('Text is required');
} else if (inputValue.length < 30) {
   // if it is, display an error message
   inputField.classList.add('is-invalid');
   inputField.setCustomValidity('Text must be at least 30
characters');
} else {
   // if it's valid, remove any error messages and submit the form
   inputField.classList.remove('is-invalid');
   inputField.setCustomValidity('');
   form.submit();
}
       });
   </script>
```

5.2. Testing

5.2.1. Test Cases for Unit Testing

The process of unit testing involves testing a "unit" of code independently and comparing the outcomes to what was anticipated. Unit tests call one or more class methods to generate observable outcomes that are automatically checked. Test-driven development is advocated by Extreme Programming (XP) proponents as a desirable and efficient method for creating software. Using this method, class behavior must first be developed in order to fulfill the test cases. The class implementation is carried out sequentially, with tests run after each change [47].

Test Case 1

Test Objectives: Test for the opening of the application and routing when clicking on buttons.

Expected Output: Successfully opened and routes working properly**.**

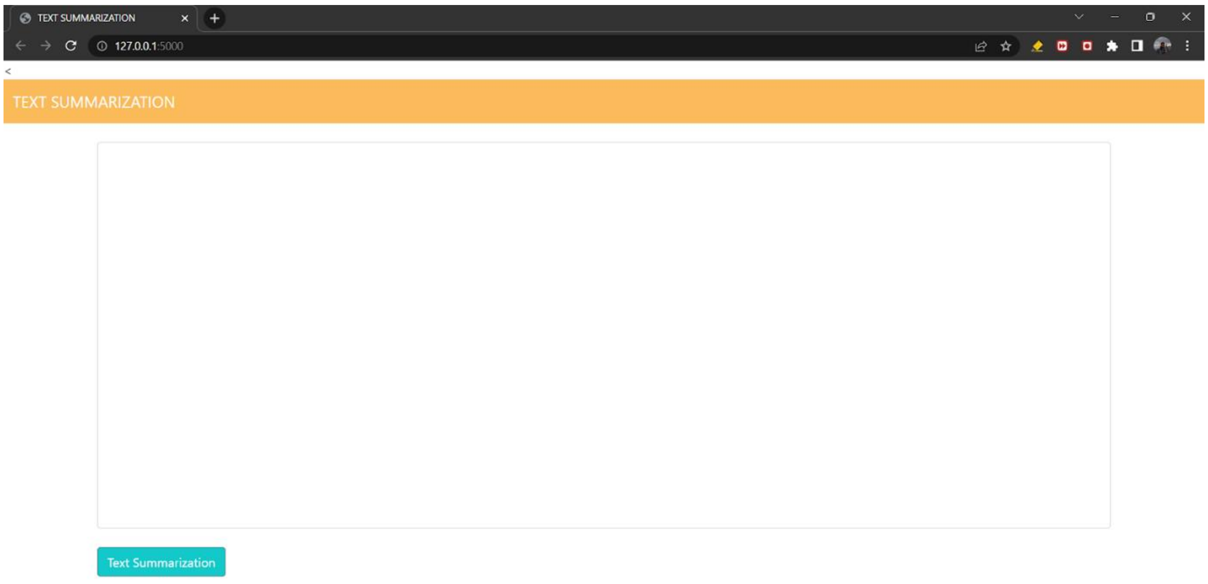
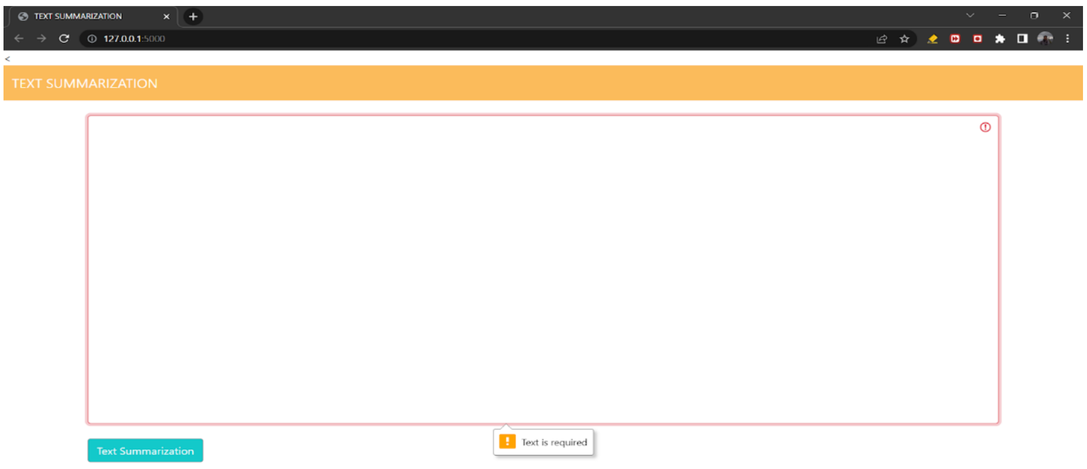


Figure 13: Test Case 1

Test Case 2

Test Objectives: Test for validating input and accepting only when text is inputted.

Expected Output: successfully shows an error message when no text is inputted.



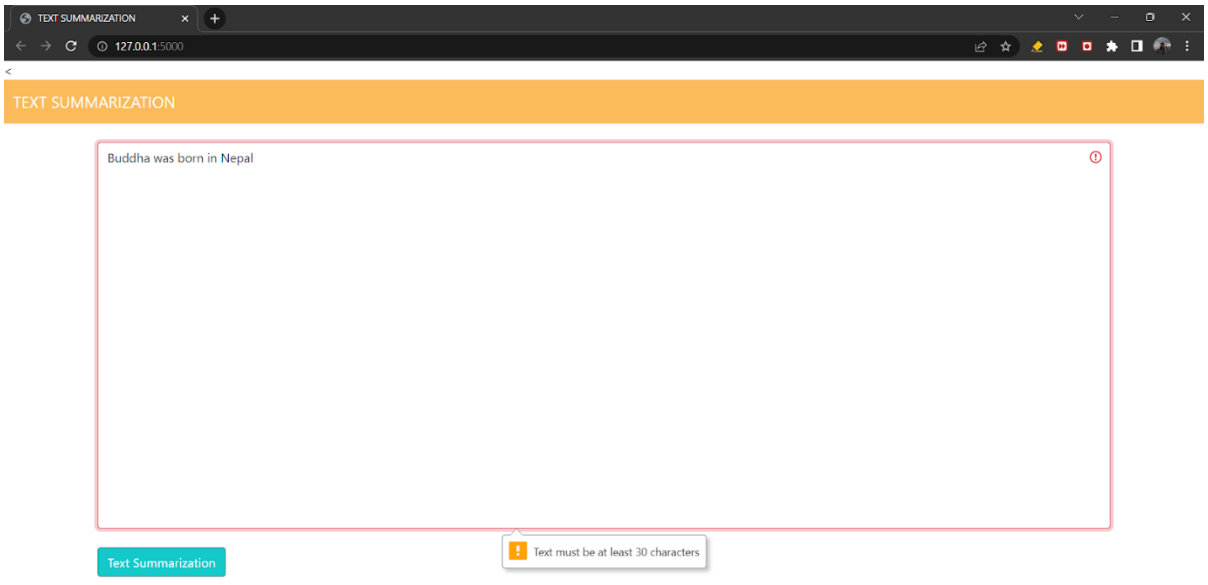
Text is required

Figure 14: Test Case 2

Test Case 3

Test Objectives: Test for validating input and accepting only when the text length is met.

Expected Output: successfully shows an error message when the text length is less than 30 characters.



Buddha was born in Nepal

Text must be at least 30 characters

Figure 15: Test Case 3

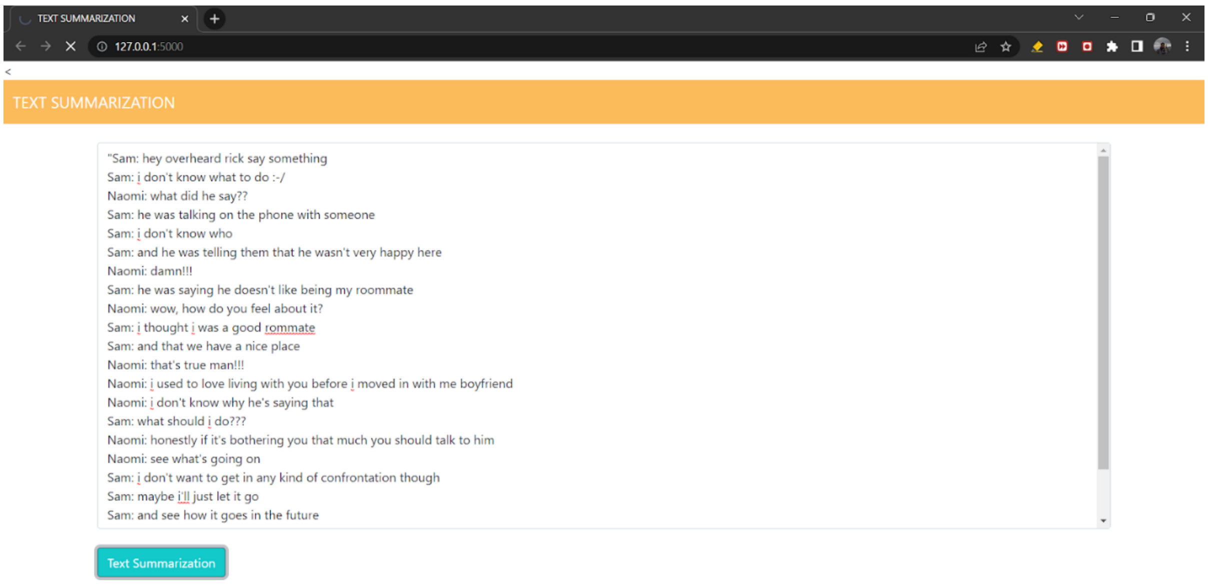
5.2.2. Test Cases for System Testing

System testing is concerned with testing an entire system based on its specifications and involves several activities such as functional testing (testing from behavioral descriptions of the system) and performance testing (response time and resource utilization). In other words, the implementation under test is compared to its intended specification [48].

Test Case

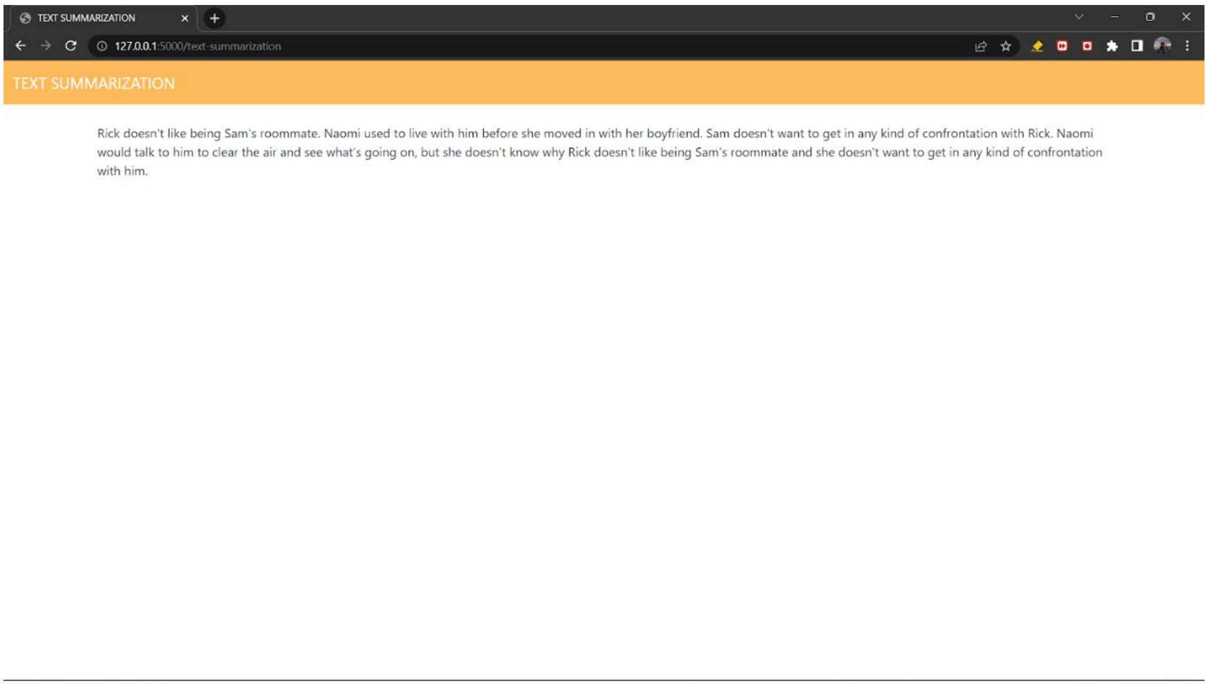
Test Objectives: Text for generating correct summary from the inputted text.

Expected Output: successfully generated a summary from the input text.



"Sam: hey overheard rick say something
Sam: i don't know what to do :-/
Naomi: what did he say??
Sam: he was talking on the phone with someone
Sam: i don't know who
Sam: and he was telling them that he wasn't very happy here
Naomi: damn!!!
Sam: he was saying he doesn't like being my roommate
Naomi: wow, how do you feel about it?
Sam: i thought i was a good rommate
Sam: and that we have a nice place
Naomi: that's true man!!!
Naomi: i used to love living with you before i moved in with me boyfriend
Naomi: i don't know why he's saying that
Sam: what should i do???
Naomi: honestly if it's bothering you that much you should talk to him
Naomi: see what's going on
Sam: i don't want to get in any kind of confrontation though
Sam: maybe i'll just let it go
Sam: and see how it goes in the future

Figure 16: Input of dialogue

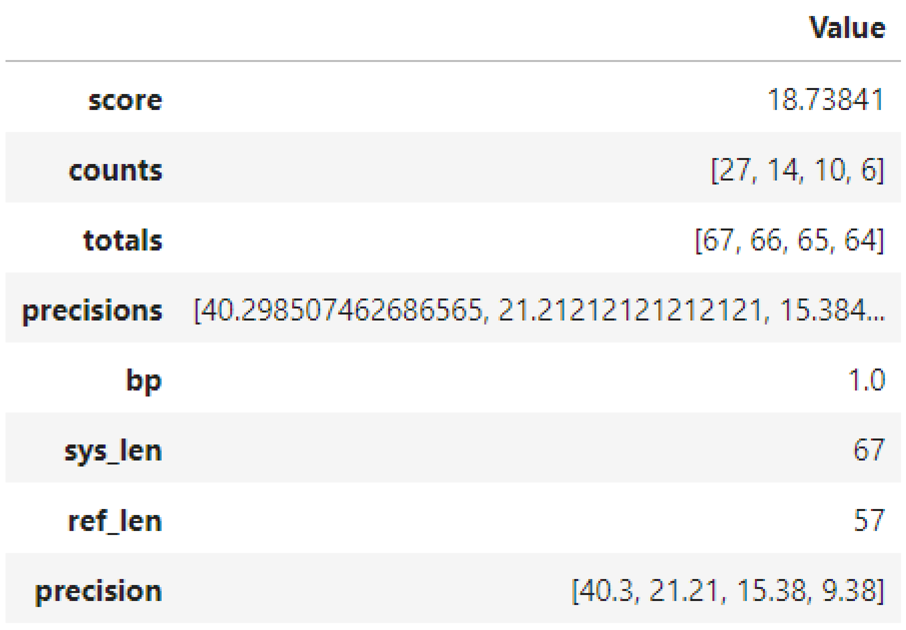


Rick doesn't like being Sam's roommate. Naomi used to live with him before she moved in with her boyfriend. Sam doesn't want to get in any kind of confrontation with Rick. Naomi would talk to him to clear the air and see what's going on, but she doesn't know why Rick doesn't like being Sam's roommate and she doesn't want to get in any kind of confrontation with him.

Figure 17: Generated Summary

5.3. Result Analysis

The authors have compared a few of the text summarization models which are baseline, gpt2, t5, bart and pegasus models for a cnn/ dailymail dataset. The authors took a sample text from the dataset and generated summaries for each of the models. Then the authors calculated SacreBLEU and ROUGE scores for each of the models.

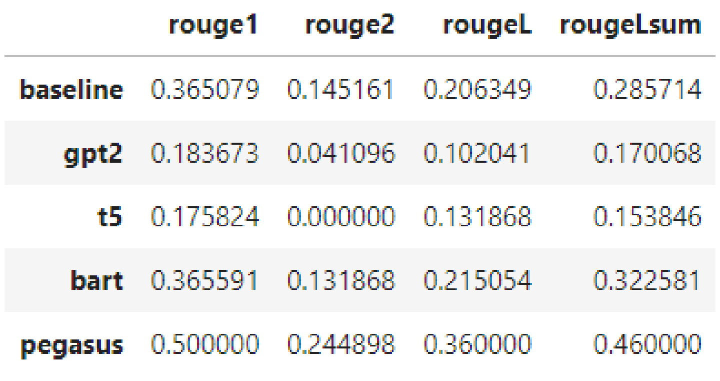


	Value
score	18.73841
counts	[27, 14, 10, 6]
totals	[67, 66, 65, 64]
precisions	[40.298507462686565, 21.2121212121212121, 15.384...
bp	1.0
sys_len	67
ref_len	57
precision	[40.3, 21.21, 15.38, 9.38]

Figure 18: Comparison between different models with SacreBLEU score

The SacreBLEU score generates the precision of the models while generating summaries. This metric has many drawbacks as it does not consider the meaning behind the sentence. Therefore a slightly advanced metric i.e. ROUGE is used to measure the score.

The authors have compared a few of the text summarization models which are baseline, gpt2, t5, Bart, and pegasus models for a CNN/ dailymail dataset. They took a sample text from the dataset and generated summaries for each of the models. Then ROUGE scores for each of the models were calculated.



	rouge1	rouge2	rougeL	rougeLsum
baseline	0.365079	0.145161	0.206349	0.285714
gpt2	0.183673	0.041096	0.102041	0.170068
t5	0.175824	0.000000	0.131868	0.153846
bart	0.365591	0.131868	0.215054	0.322581
pegasus	0.500000	0.244898	0.360000	0.460000

Figure 19: Comparison between different models with ROUGE score

From the table above, it is clear that the pegasus model has generated a summary with a high ROUGE score.

Now, the authors tried generating a summary for a custom dataset (samsum dataset) using the previous model. Samsum dataset is a dataset containing dialogues between people. From the histogram below, it can be observed that the structure of a news article and a conversation is different. Thus, fine-tuning and training the model which references the custom dataset was needed.

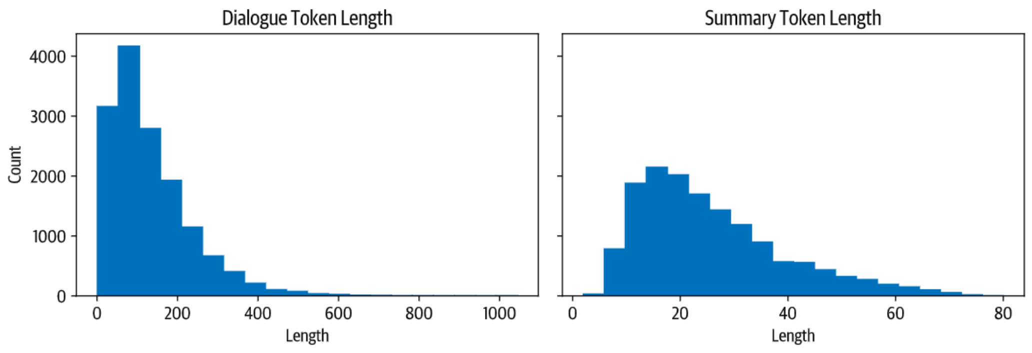
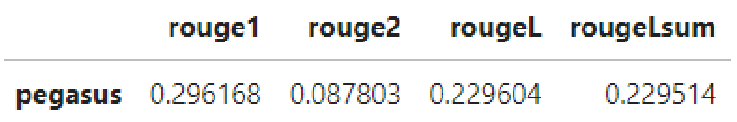
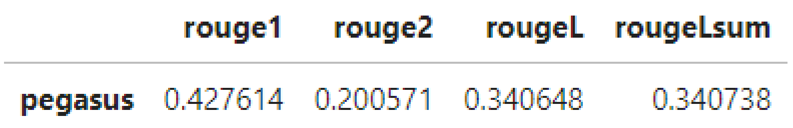


Figure 20: Histogram of dialogue and summary



	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.296168	0.087803	0.229604	0.229514

Figure 21: ROUGE score before fine-tuning



	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.427614	0.200571	0.340648	0.340738

Figure 22: ROUGE score after fine-tuning

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS

6.1. Conclusion

Text summarization is a useful technique that can save time and effort by summarizing large volumes of text into shorter summaries. It increases efficiency, improves accessibility for individuals who have difficulty reading or comprehending large volumes of text, and enables better decision-making. However, it also has some disadvantages, such as loss of information, limited accuracy, difficulty with complex texts, and dependence on the quality of the dataset used to train the summarization model. Therefore, it is important to carefully consider the advantages and disadvantages before implementing a text summarization project. When used appropriately, text summarization can be a powerful tool that can greatly benefit individuals and organizations.

In summary, Pegasus is a cutting-edge text summarization model that has been demonstrated to perform well on a variety of datasets. The model can be adjusted to provide excellent summaries that accurately represent the key details in the original text when used with a unique dataset.

However, the success of text summarization using Pegasus for a custom dataset depends on several factors, including the quality and size of the dataset, the complexity and diversity of the text, and the training parameters used. It is important to carefully preprocess and clean the dataset to ensure that the model is trained on high-quality data. Overall, Pegasus is a powerful tool for text summarization, but it is important to use it in conjunction with other techniques such as human review and domain-specific knowledge to ensure that the generated summaries are accurate and relevant to the intended audience.

Text summarization has many practical applications in various fields, including journalism, business, education, and research. In journalism, text summarization can be used to quickly generate brief news summaries for readers who want to stay informed but have limited time. In the business world, text summarization can help to analyze and summarize large volumes of financial reports, legal documents, and customer feedback. In education, text summarization can be used to simplify complex academic papers and textbooks, making the material more accessible to students. In research, text summarization can assist researchers in quickly identifying relevant studies and extracting key information from scientific papers. Overall, text summarization has many practical applications in streamlining information processing and facilitating decision-making across various fields.

6.2. Future Recommendations

- For the Nepali language: Developing a text summarization model that can summarize Nepali text and articles. To do so collection and labeling of datasets are required.
- Improved accuracy: Developing text summarization models that can accurately capture the nuances of the original text is essential. Future research should focus on improving the accuracy of summarization models, especially for complex texts and diverse languages.
- Personalization: Creating text summarization models that can be personalized to the specific needs and preferences of the user can enhance the usefulness and effectiveness of the summarization process.
- Multi-modal summarization: Developing summarization models that can combine text with other modalities, such as images and videos, can improve the quality of the summary by providing a more comprehensive and holistic understanding of the information.

CHAPTER 7: REFERENCES

[1] F. Kiyani and O. Tas, “A survey automatic text summarization,” *Pressacademia*, vol. 5, no. 1, pp. 205–213, Jun. 2017, doi: 10.17261/PRESSACADEMIA.2017.591.

[2] N. Moratanch and S. Chitrakala, “A survey on abstractive text summarization,” *Proc. IEEE Int. Conf. Circuit, Power Comput. Technol. ICCPCT 2016*, Aug. 2016, doi: 10.1109/ICCPCT.2016.7530193.

[3] M. Allahyari *et al.*, “Text Summarization Techniques: A Brief Survey,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 10, Jul. 2017, doi: 10.48550/arxiv.1707.02268.

[4] “An Introduction to Neural Networks - James A. Anderson - Google Books.” https://books.google.com.np/books?hl=en&lr=&id=_ib4vPdB76gC&oi=fnd&pg=PP11&dq=an+introduction+to+neural+networks&ots=WcfwN7CMZf&sig=gG0bsjl6kIFPYZ8nDfEC1GFBuaU&redir_esc=y#v=onepage&q=an introduction to neural networks&f=false (accessed Feb. 01, 2023).

[5] A. Graves, “Generating Sequences With Recurrent Neural Networks,” Aug. 2013, doi: 10.48550/arxiv.1308.0850.

[6] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013, doi: 10.1109/TPAMI.2013.50.

[7] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[8] S. Song, H. Huang, and T. Ruan, “Abstractive text summarization using LSTM-CNN based deep learning,” *Multimed. Tools Appl.*, vol. 78, no. 1, pp. 857–875, Jan. 2019, doi: 10.1007/S11042-018-5749-3.

[9] H. Aliakbarpour, M. T. Manzuri, and A. M. Rahmani, “Improving the readability and saliency of abstractive text summarization using combination of deep neural networks equipped with auxiliary attention mechanism,” *J. Supercomput.*, vol. 78, no. 2, pp. 2528–2555, Feb. 2022, doi: 10.1007/S11227-021-03950-X.

[10] A. Vaswani *et al.*, “Attention Is All You Need,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, pp. 5999–6009, Jun. 2017, Accessed: Apr. 23, 2023. [Online]. Available: https://arxiv.org/abs/1706.03762v5

[11] C. Li, W. Xu, S. Li, and S. Gao, “Guiding Generation for Abstractive Text Summarization Based on Key Information Guide Network,” *NAACL HLT 2018 -*

2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf., vol. 2, pp. 55–60, 2018, doi: 10.18653/V1/N18-2009.

[12] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, “Neural Abstractive Text Sum-marization with Sequence-to-Sequence Models,” *ACM/IMS Trans. Data Sci*, vol. 2, no. 1, 2020, doi: 10.1145/3419106.

[13] J. Cheng and M. Lapata, “Neural Summarization by Extracting Sentences and Words”.

[14] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, “PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization,” p. 119, 2020.

[15] J. Pilault, R. Li, S. Subramanian, C. Pal, and P. Montreal, “On Extractive and Abstractive Neural Document Summarization with Transformer Language Models,” pp. 9308–9319.

[16] J. Li, C. Zhang, X. Chen, Y. Cao, P. Liao, and P. Zhang, “Abstractive Text Summarization with Multi-Head Attention,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2019-July, no. July, pp. 1–8, 2019, doi: 10.1109/IJCNN.2019.8851885.

[17] R. Nallapati, F. Zhai, and B. Zhou, “SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents,” *31st AAAI Conf. Artif. Intell. AAAI 2017*, no. c, pp. 3075–3081, 2017, doi: 10.1609/aaai.v31i1.10958.

[18] Y. Liu and M. Lapata, “Text Summarization with Pretrained Encoders,” pp. 3730–3740, Accessed: Feb. 06, 2023. [Online]. Available: https://github.com/

[19] P. Li, W. Lam, B. Lidong, and Z. Wang, “Deep Recurrent Generative Decoder for Abstractive Text Summarization *”.

[20] J. Tan, X. Wan, and J. Xiao, “Abstractive Document Summarization with a Graph-Based Attentional Neural Model,” pp. 1171–1181, doi: 10.18653/v1/P17-1108.

[21] X. Duan, H. Yu, M. Yin, M. Zhang, W. Luo, and Y. Zhang, “Contrastive Attention Mechanism for Abstractive Sentence Summarization,” pp. 3044–3053, Accessed: Mar. 09, 2023. [Online]. Available: https://github.com/travel-go/

[22] X. Liu and L. Xv, “Abstract summarization based on the combination of transformer and LSTM,” *Proc. - 2019 Int. Conf. Intell. Comput. Autom. Syst. ICICAS 2019*, pp. 923–927, Dec. 2019, doi: 10.1109/ICICAS48597.2019.00199.

[23] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li, “Generative Adversarial Network for Abstractive Text Summarization,” *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, pp. 8109–8110, Apr. 2018, doi: 10.1609/AAAI.V32I1.12141.

[24] K. Yao, L. Zhang, D. Du, T. Luo, L. Tao, and Y. Wu, "Dual encoding for abstractive text summarization," *IEEE Trans. Cybern.*, vol. 50, no. 3, pp. 985–996, 2020, doi: 10.1109/TCYB.2018.2876317.

[25] W. Kry´scí, K. Nski, B. Mccann, C. Xiong, and R. Socher, "Evaluating the Factual Consistency of Abstractive Text Summarization".

[26] T. Tsonkov, G. Lazarova, V. Zmiycharov, I. K.- ERIS, and undefined 2021, "A Comparative Study of Extractive and Abstractive Approaches for Automatic Text Summarization on Scientific Texts.," *ceur-ws.org*, 2021, Accessed: Mar. 09, 2023. [Online]. Available: https://ceur-ws.org/Vol-3061/ERIS_2021-art03(sh).pdf

[27] W. Li, X. Xiao, Y. Lyu, and Y. Wang, "Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling," *Proc. 2018 Conf. Empir. Methods Nat. Lang. Process. EMNLP 2018*, pp. 1787–1796, 2018, doi: 10.18653/V1/D18-1205.

[28] T. Barrier, "Systems Analysis," *Encycl. Inf. Syst.*, pp. 345–349, 2003, doi: 10.1016/B0-12-227240-4/00177-5.

[29] J. T. Catanio, "Requirements analysis: A review," *Adv. Syst. Comput. Sci. Softw. Eng. - Proc. SCSS 2005*, pp. 411–418, 2006, doi: 10.1007/1-4020-5263-4_64/COVER.

[30] "What is functional requirements? | Definition from TechTarget." https://www.techtarget.com/whatis/definition/functional-requirements (accessed Mar. 10, 2023).

[31] "UML Use Case Diagram - Javatpoint." https://www.javatpoint.com/uml-use-case-diagram (accessed Mar. 10, 2023).

[32] S. Paradkar, "Mastering Non-Functional Requirements Analysis, architecture, and assessment," p. 672, 2017, Accessed: Mar. 10, 2023. [Online]. Available: https://www.packtpub.com/product/mastering-non-functional-requirements/9781788299237

[33] S. Skidmore, "The Feasibility Study," *Introd. Syst. Anal.*, pp. 25–34, 1997, doi: 10.1007/978-1-349-14672-7_3.

[34] Q. Li and Y.-L. Chen, "Data Flow Diagram," *Model. Anal. Enterp. Inf. Syst.*, pp. 85–97, 2009, doi: 10.1007/978-3-540-89556-5_4.

[35] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, Sep. 2021, doi: 10.1016/J.NEUCOM.2021.03.091.

[36] 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, Feb. 2016, doi: 10.48550/arxiv.1602.06023.

[37] S. Kim, “Using Pre-Trained Transformer for Better Lay Summarization,” pp. 328–335, Nov. 2020, doi: 10.18653/V1/2020.SDP-1.38.

[38] M. Barbella and G. Tortora, “Rouge Metric Evaluation for Text Summarization Techniques,” *SSRN Electron. J.*, May 2022, doi: 10.2139/SSRN.4120317.

[39] M. Post, “A Call for Clarity in Reporting BLEU Scores,” *WMT 2018 - 3rd Conf. Mach. Transl. Proc. Conf.*, vol. 1, pp. 186–191, Apr. 2018, doi: 10.48550/arxiv.1804.08771.

[40] “Get started | PyCharm Documentation.” https://www.jetbrains.com/help/pycharm/quick-start-guide.html#meet (accessed Mar. 10, 2023).

[41] “What is MS Word? - Basics, Uses, Features & Questions.” https://byjus.com/govt-exams/microsoft-word/ (accessed Mar. 10, 2023).

[42] “What is Google Drive - Definition, meaning and examples.” https://www.arimetrics.com/en/digital-glossary/google-drive (accessed Mar. 10, 2023).

[43] “What is Google Colab?” https://www.androidpolice.com/google-colab-explainer/ (accessed Mar. 10, 2023).

[44] “What is Draw.io?” https://www.computerhope.com/jargon/d/drawio.htm (accessed Mar. 10, 2023).

[45] “What is Google Chrome browser? | Definition from TechTarget.” https://www.techtarget.com/searchmobilecomputing/definition/Google-Chrome-browser (accessed Mar. 10, 2023).

[46] “How Does Grammarly Work? A Comprehensive Guide - Financesonline.com.” https://financesonline.com/how-does-grammarly-work-a-comprehensive-guide/ (accessed Mar. 11, 2023).

[47] M. Olan, “UNIT TESTING: TEST EARLY, TEST OFTEN *,” 2003, Accessed: Mar. 10, 2023. [Online]. Available: https://www.researchgate.net/publication/255673967

[48] L. Briand and Y. Labiche, “A UML-Based Approach to System Testing,” *Softw. Syst. Model.*, vol. 1, no. 1, pp. 10–42, Sep. 2002, doi: 10.1007/S10270-002-0004-8.

APPENDICES

Abstractive Text Summarization Using Transformer Model

ORIGINALITY REPORT

5%	4%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	1%
2	www.aclweb.org Internet Source	1%
3	uwe-repository.worktribe.com Internet Source	1%
4	Submitted to University of Wales Institute, Cardiff Student Paper	1%
5	Submitted to Durban University of Technology Student Paper	1%
6	Submitted to King's Own Institute Student Paper	1%
7	whitepapers.hackerjournals.com Internet Source	1%

Exclude quotes	On	Exclude matches	< 1%