Design and Development of Text Summarization

A

MINOR PROJECT-II REPORT

Submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

By

GROUP NO. 8

Sanskar Agrawal	0187AD211036
Shivam Pathak	0187AD211037
Aryan Sharma	0187AD211010

Under the guidance of

Dr. Vasima Khan

(ASSOCIATE PROFESSOR)

JUNE-2024

Department of Artificial Intelligence & Data Science
Sagar Institute of Science & Technology (SISTec)
Bhopal (M.P.)

Approved by AICTE, New Delhi & Govt. of M.P.

Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

Sagar Institute of Science & Technology (SISTec), Bhopal

Department of Artificial Intelligence & Data Science

Bhopal (M.P.)

JUNE-2024

CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Minor Project-II Report entitled **Design & Development of Abstractive Text Summarization,** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology*** in ***Artificial Intelligence & Data Science*** and submitted to the Department of Computer Science & Engineering, *Sagar Institute of Science & Technology (SISTec)***,** Bhopal (M.P.) is an authentic record of my own work carried out during the period from Jan-2024 to June-2024 under the supervision of **Dr. Vasima Khan (Associate Professor).** The content presented in this project has not been submitted by me for the award of any other degree elsewhere.

Sanskar Agrawal	Shivam Pathak	Aryan Sharma
0187AD211036	0187AD211037	0187AD211010

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

Date:

Dr. Vasima Khan	**Dr. Vasima Khan**	**Dr. D.K. Rajoriya**
Associate Professor	**HOD**	**Principal**

ABSTRACT

The ever-growing deluge of text data necessitates innovative methods for knowledge extraction and comprehension. This project tackles this challenge by designing and developing a text summarization system powered by a Pegasus model, a state-of-the-art deep learning approach for abstractive summarization. Unlike traditional methods that simply stitch together existing sentences, Pegasus excels at generating novel summaries that capture the essence of the source text, providing a deeper understanding of the content. To achieve this, the project prioritizes both functional and non-functional requirements, ensuring flexibility in text input (user-provided text, text files, or web documents), core summarization functionality with user control over summary length, and a user-friendly interface. Additionally, non-functional requirements emphasize the accuracy and clarity of generated summaries, alongside efficient processing and a user-friendly interface. To meet these goals, the project employs a Pegasus model meticulously trained on a well-curated dataset. The data undergoes preprocessing techniques like tokenization (breaking text into manageable units) and truncation (limiting text length) to optimize it for the model. During training, the project leverages advanced functionalities offered by deep learning libraries to ensure efficient learning and performance. The culmination of this project is a functional text summarization system capable of generating summaries for unseen text inputs. This demonstrates the model's ability to learn and generalize its knowledge beyond the training data. Future work can explore further optimization of hyperparameters to enhance performance and delve into real-world applications, such as summarizing news articles or research papers. Ultimately, this project aspires to empower users with a powerful tool that improves information access and comprehension in the face of an ever-expanding ocean of text data.

The "Automatic Text Summarization with Pegasus Model" project empowers users across various fields to efficiently navigate the ever-growing volume of textual information. This system offers automatic generation of concise summaries that capture the essence of lengthy texts, improving information access and comprehension. Furthermore, the project provides valuable insights into the capabilities of deep learning models for text summarization and the potential applications in real-world scenarios. It also serves as a learning resource for anyone interested in exploring data pre-processing, model training, and evaluation techniques within the field of Natural Language Processing (NLP).

ACKNOWLEGMENT

We would like to extend our heartfelt gratitude to various individuals and institutions who have played a significant role in the successful completion of our project, "Design & Development of Abstractive Text Summarization Model."

First and foremost, we wish to express our sincere appreciation to our project guide, **Dr. Vasima Khan**, for their unwavering support, invaluable guidance, and insightful feedback throughout the course of this project.

We are deeply grateful to **Sagar Institute of Science & Technology** for providing us with the necessary resources, facilities, and a conducive environment for research and learning. This project would not have been possible without the support and opportunities offered by our esteemed institution.

Our heartfelt thanks go to **Dr. D.K. Rajoriya**, Principal of Sagar Institute of Science & Technology, for granting us the opportunity to undertake this project. We also extend our appreciation to **Dr. Swati Saxena,** Vice Principal of Sagar Institute of Science & Technology, for their support and encouragement throughout our endeavor.

We acknowledge the invaluable assistance and support provided by **Dr. Vasima Khan**, Head of the **Department of Artificial Intelligence & Data Science,** who offered valuable insights and encouragement at various stages of our project.

Special recognition goes to **Prof. Abhuday Tripathi**, Assistant Professor in the Department of Artificial Intelligence & Data Science, for their unwavering support and readiness to address our queries and concerns. Their academic expertise and commitment to our project were truly commendable.

Name	**Enrollment Number**	**Signature**
Sanskar Agrawal	0187AD211036	
Shivam Pathak	0187AD211037	
Aryan Sharma	0187AD211010	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
RNN	Recurrent Neural Network
NLP	Natural Language Processing
LSTM	Long Short-Term Memory

Chapter 1
Introduction

CHAPTER-1 INTRODUCTION

1.1 ABOUT THE PROJECT

The digital age has brought a deluge of information, making it difficult to efficiently extract key points and gain valuable insights. Automatic text summarization emerges as a powerful tool in this fight against information overload. It condenses lengthy text into concise summaries, retaining the core meaning and facilitating information access.

This project delves into the design and development of an automatic text summarization system leveraging the capabilities of a Pegasus model. Unlike traditional methods that simply select existing sentences, Pegasus is a cutting-edge deep learning approach that excels at abstractive summarization. It generates novel summaries that capture the essence of the source text, providing a deeper understanding of the content.

The project prioritizes both functional and non-functional requirements to ensure a user-friendly and effective summarization tool. Functional requirements focus on flexibility in text input (user-provided text, text files, or web documents if applicable), core summarization functionality (automatically generating concise summaries), and user control over summary length. Non-functional requirements emphasize accuracy and clarity of generated summaries, efficiency of the summarization process, and a user-friendly interface for smooth interaction.

To achieve these goals, the project employs a meticulously trained Pegasus model on a well-curated dataset. The data undergoes pre-processing techniques like tokenization and truncation to optimize it for the model. During training, the project leverages advanced functionalities from deep learning libraries for efficient learning and performance.

The project culminates in a functional text summarization system capable of generating summaries for unseen text inputs. This demonstrates the model's ability to learn and generalize its knowledge beyond the training data, making it adaptable to new information and real-world scenarios.

While the project establishes a core text summarization system, future work can explore further enhancements and delve into real-world applications. This could involve hyperparameter optimization for improved accuracy, integration with real-world tools for easy access, and even expanding the model's capabilities to handle multiple languages or be customized for specific domains. By tackling information overload and empowering users with efficient text summarization tools, this project contributes to improved information access and comprehension in our ever-expanding digital world.

- **PROJECT OBJECTIVES**

The primary objectives of this project are as follows:

- ♦ **Effective Text Summarization:** Develop a system capable of generating concise and informative summaries that capture the essence of lengthy text inputs. This ensures accurate representation of key ideas and information while maintaining clarity and avoiding redundancy.
- ♦ **User-Friendly Text Input:** Design the system to accept user-provided text as input. This simplifies the summarization process and caters to individual needs.
- ♦ **Prioritize Accuracy and Clarity:** Ensure the generated summaries are accurate and faithful representations of the original text, capturing key points and avoiding factual errors. Additionally, clarity and conciseness are crucial for user understanding.
- ♦ **Efficient Summarization Process:** Develop a system that generates summaries efficiently, particularly for longer texts. This minimizes waiting times and improves user experience.
- ♦ **Intuitive User Interface:** Create an intuitive user interface that allows users to easily input text and access generated summaries. This fosters seamless interaction for the core summarization functionality.

1.2 SIGNIFICANCE OF ABSTRACTIVE TEXT SUMMARIZATION

In today's data deluge, information crashes over us like a relentless wave. Your text summarization project cuts through the noise like a lighthouse, guiding users towards the key points hidden within mountains of text. It empowers users by automatically generating concise summaries, saving them valuable time and effort – time that could be spent reading novels, exploring hobbies, or simply taking a well-deserved break.

This translates to boosted productivity across various fields. Imagine researchers who can analyze five times the number of papers in a day, students who can grasp complex topics during a single study session, and busy professionals who can make informed decisions based on summarized reports within minutes. The saved time allows them to focus on higher-level tasks requiring human expertise, such as creative problem-solving, strategic planning, or client interaction.

The benefits extend far beyond individual users. Summarization can democratize access to information, fostering a more inclusive information landscape. Imagine a world where visually impaired individuals can access news articles with the aid of text-to-speech tools, or where those with dyslexia can overcome reading difficulties by listening to clear, concise summaries. This empowers everyone to participate meaningfully in our knowledge-driven society, regardless of background or ability.

Chapter 2
Software & Hardware Requirements

CHAPTER-2
SOFTWARE & HARDWARE REQUIREMENTS

2.1 SOFTWARE REQUIREMENTS

The successful development and deployment of the "Design and Development of Abstractive Text Summarization " project necessitate the utilization of specific software tools and technologies.This section outlines the software requirements vital to the project's execution.

- **Operating System:** Windows 10+
- **Programming Language:** Python 3.12.0
- **Framework:** Flask
- **Libraries:** PyTorch , datasets , transformers

2.1 HARDWARE REQUIREMENTS

The "Design and Development of A " project demands specific hardware resources to ensure its smooth operation. The hardware requirements include:

CPU	:	Quad-core Intel Core i5 or higher
RAM	:	16 GB or higher
Hard Disk	:	512 GB SSD or higher
Graphic card	:	NVIDIA GeForce RTX series

Client-Side Requirements : Browser (Any Compatible browser device)

Chapter 3
Problem Description

CHAPTER-3
PROBLEM DESCRIPTION

The digital age is a double-edged sword. While information is more accessible than ever, the sheer volume creates a new challenge: information overload. We're bombarded with text from research papers to social media posts, making it difficult to efficiently extract key points and gain valuable insights. Imagine a researcher struggling to keep up with the ever-increasing number of publications in their field, or a student facing a mountain of text to understand a complex topic. Even staying informed on current events becomes a chore with the constant influx of news articles.

This information overload hinders our ability to process information effectively. Sifting through lengthy documents to find relevant information is time-consuming, making it difficult to grasp complex topics, stay informed, and make well-informed decisions.

The problem cuts across various fields. Researchers drown in a sea of publications, students wrestle with textbooks and articles, and professionals grapple with reports and industry trends – all due to the sheer volume of text they encounter. Even the general public faces hurdles in keeping up with the news cycle and accessing critical information.

Chapter 4
Literature Survey

CHAPTER-4 LITERATURE SURVEY

This survey explores the code implementation aspects of training a transformer-based model for text summarization. Traditionally, techniques like tokenization (provided by NLTK [1]) and Long Short-Term Memory (LSTM) networks [2] were used for text processing and summarization. However, transformers [3], introduced by Vaswani et al., revolutionized the field due to their ability to capture long-range dependencies in text.

The Transformers library [4] provides pre-trained models and functionalities for various NLP tasks, including text summarization. This code leverages the AutoModelForSeq2SeqLM class to load the pre-trained PEGASUS model [6] for summarization. The Datasets library [5] simplifies loading and processing large datasets, while DataCollatorForSeq2Seq and TrainingArguments classes from Transformers [4] prepare data for training and define hyperparameters, respectively. The core training process is handled by the Trainer class, which utilizes the model, arguments, tokenizer, data collator, and training/evaluation datasets.

Pre-trained models like PEGASUS are particularly well-suited for text summarization due to their ability to handle both encoder and decoder functionalities within the transformer architecture. However, other options like BART [7], T5 [8], and BERT [9] can also be explored depending on specific needs and resources. Text data is converted into numerical representations using a tokenizer, with options like Punkt from NLTK [1] for sentence segmentation within the preprocessing pipeline. Random sampling is employed to reduce training dataset size for efficiency.

Fine-tuning the pre-trained model on a specific summarization dataset (like CNN/Daily Mail) leverages transfer learning [10] to adapt the model's knowledge to the task. The training loop iterates through batches of data, updates model weights, and evaluates the model periodically. While ROUGE score [11] is a common metric, exploring additional evaluation methods like human evaluation or BLEU score can provide a more comprehensive understanding of summarization quality.

Finally, the trained model and tokenizer can be saved for future use, and frameworks like Flask [12] can be used to deploy the model as a web application for real-time text summarization.

Chapter 5
Software Requirement Specification

CHAPTER-5
SOFTWARE REUIREMENTS SPECIFICATION

This chapter outlines the functional and non-functional requirements of the "Design and Development of Abstractive Text Summarization." Subsequent chapters will delve into the software design and development, providing a detailed understanding of how these requirements will be realized.

5.1 FUNCTIONAL REQUIRMENTS

The "Design and Development of Abstractive Text Summarization " project encompasses a range of functional and non-functional requirements crucial for the project's success.

- **Text Input:** The system should accept text input from user-provided text and perform summarization on it.
- **Text Summarization:** The core functionality is to automatically generate concise summaries that capture the essential meaning of the input text.

5.1 NON- FUNCTIONAL REQUIREMENTS

In addition to functional requirements, the project has non-functional requirements that focus on the system's quality, performance, and usability.

- **Accuracy:** The generated summaries should be accurate and faithful representations of the original text, capturing key points and avoiding factual errors.
- **Clarity and Conciseness:** Summaries should be clear, concise, and easy to understand, avoiding redundancy or excessive detail.
- **Efficiency:** The summarization process should be efficient, generating summaries in a reasonable timeframe.
- **User-Friendliness:** The system should have a user-friendly interface that allows users to easily input text, control summary length, and access generated summaries.

Chapter 6
Software Design

CHAPTER-6
SOFTWARE DESIGN

6.1 OVERVIEW

This chapter outlines the software design for the "Design and Development of Abstractive Text Summarization" project.

6.2 SYSTEM ARCHITECTURE

The software architecture consists of data input and preprocessing, a Transformer model, a Encoder and Decoder.

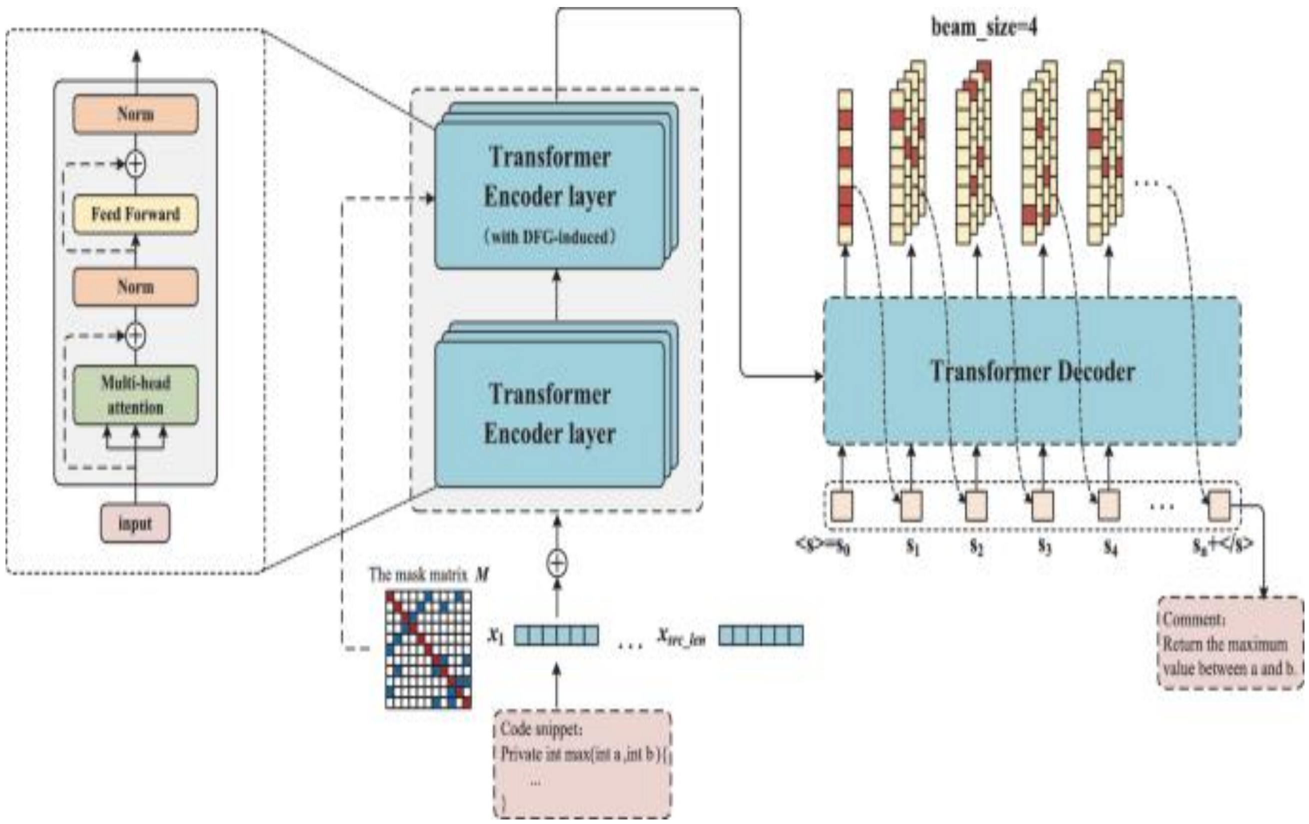


Fig (6.1): System Architecture

6.3 DATA FLOW

The data flow begins with text input and tokenization, followed by text encoding & then decoding, summarization and provide summarized text. This structure ensures accurate text summarization.

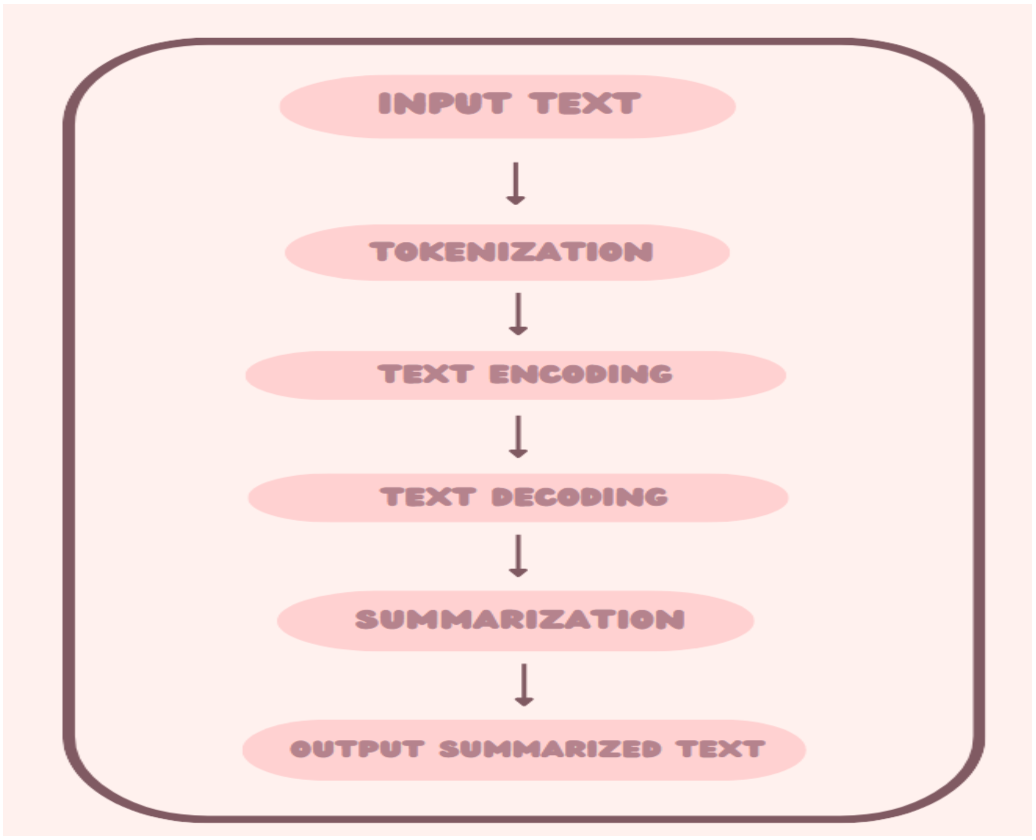


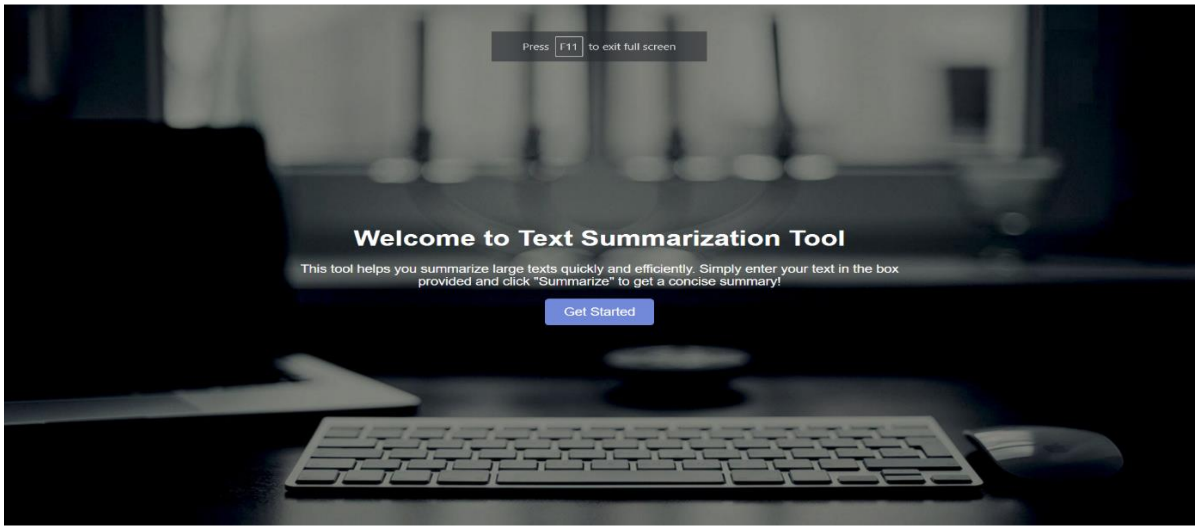
Fig (6.2): Data Flow Diagram

Chapter 7
Output Screens

CHAPTER-7
OUTPUT SCREENS

This chapter presents visual representations of the user interface for the "Design and Development of Abstractive Text Summarization Model.

Home Page: The entry point for users, offering navigation option to get started.

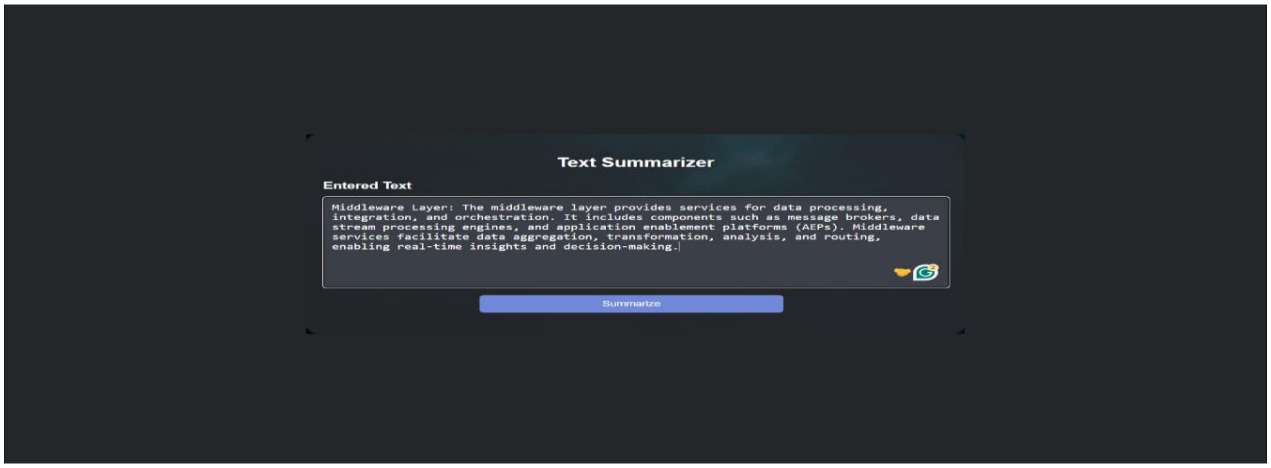


Welcome to Text Summarization Tool

This tool helps you summarize large texts quickly and efficiently. Simply enter your text in the box provided and click "Summarize" to get a concise summary!

(Fig 7.1)

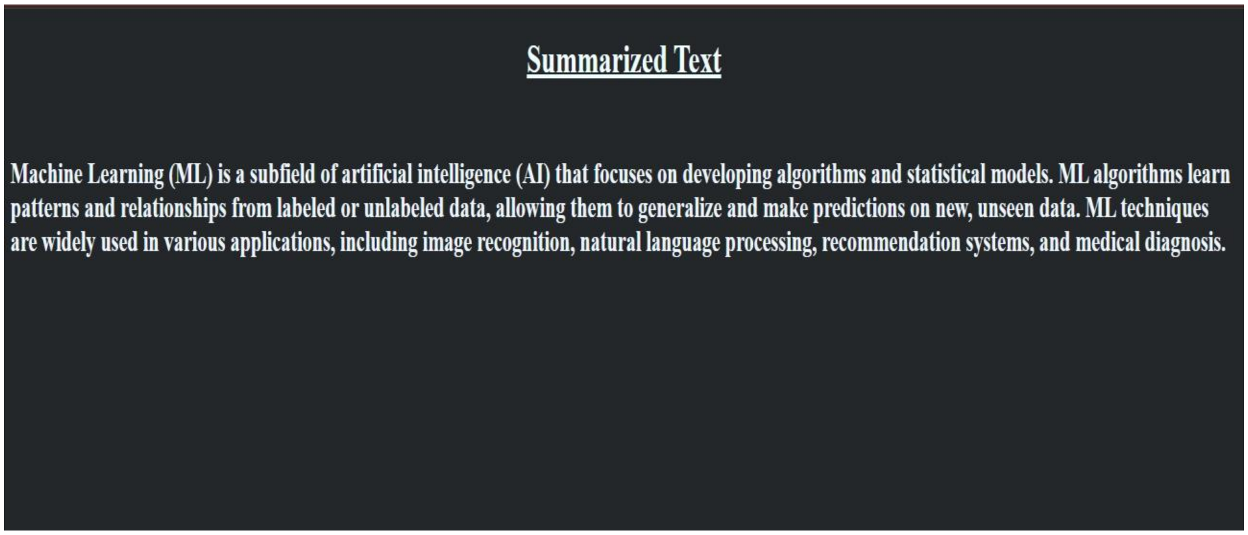
Input Interface: Allows users to input text.



Middleware Layer: The middleware layer provides services for data processing, integration, and orchestration. It includes components such as message brokers, data stream processing engines, and application enablement platforms (AEPs). Middleware services facilitate data aggregation, transformation, analysis, and routing, enabling real-time insights and decision-making.

(Fig 7.2)

Output Interface: Displays accurate Summarization of the text input.



Machine Learning (ML) is a subfield of artificial intelligence (AI) that focuses on developing algorithms and statistical models. ML algorithms learn patterns and relationships from labeled or unlabeled data, allowing them to generalize and make predictions on new, unseen data. ML techniques are widely used in various applications, including image recognition, natural language processing, recommendation systems, and medical diagnosis.

(Fig 7.3)

Chapter 8
Model Development

CHAPTER-8
Model Development

8.1 MODEL IMPLEMENTATION

# Install prerequired Libraries

```
!pip install transformers[sentencepiece] datasets sacrebleu rouge_score py7zr -q
!pip install --upgrade accelerate
!pip uninstall -y transformers accelerate
!pip install transformers accelerate
```

# Import Libraries

```
from datasets import load_dataset
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
import torch

device = "cuda" if torch.cuda.is_available() else "cpu"
```

# Import the Pre-Trained Model's

```
tokenizer = AutoTokenizer.from_pretrained('google/pegasus-xsum')
model_peg = AutoModelForSeq2SeqLM.from_pretrained('google/pegasus-xsum').to(device)
```

# Import the Dataset

```
dataset = load_dataset('cnn_dailymail','3.0.0')
```

# Preprocess The Dataset

```
from datasets import DatasetDict import random

original_train_data = dataset['train']
total_rows = len(original_train_data)
desired_size = 2000
indices_to_keep = random.sample(range(total_rows), desired_size)
dataset['train'] = original_train_data.select(indices_to_keep)


original_test_data = dataset['test']
total_rows = len(original_train_data)
desired_size = 2000
indices_to_keep = random.sample(range(total_rows), desired_size)
dataset['test'] = original_test_data.select(indices_to_keep)
```

```
original_validation_data = dataset[‘validation’]
total_rows = len(original_train_data)
desired_size = 2000
indices_to_keep = random.sample(range(total_rows), desired_size)
dataset[‘validation’] = original_validation_data (indices_to_keep)
```

# Dataset Format

```
dataset

DatasetDict ({
train: Dataset({ features: ['article', 'highlights', 'id'], num_rows: 2000 })
validation: Dataset({ features: ['article', 'highlights', 'id'], num_rows: 2000 })
test: Dataset({ features: ['article', 'highlights', 'id'], num_rows: 2000 })
})

def convert_data(batch):
        input_encoding = tokenizer(batch['article'], max_length=1024, truncation = True )

        with tokenizer.as_target_tokenizer():
                target_encoding = tokenizer(batch['highlights'], max_length=200, truncation = True )

        return {
                'input_ids' : input_encoding['input_ids'],
                'attention_mask' : input_encoding['attention_mask'],
                'labels' : target_encoding['input_ids']
                }

data_samsum_pt = dataset.map(convert_data, batched= True )

from transformers import DataCollatorForSeq2Seq
seq_data_collater = DataCollatorForSeq2Seq(tokenizer, model =model_peg )

from transformers import TrainingArguments, Trainer

training_arg = TrainingArguments(
        output_dir = 'pegasus-cnn_dailymail', num_train_epochs=4, 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_peg, args=training_arg, tokenizer = tokenizer,
                    data_collator= seq_data_collater, train_dataset=data_samsum_pt['test'],
                    eval_dataset=data_samsum_pt['validation']
                    )
```

# Model Training

```
trainer.train()
```

# Model Evaluation

```
trainer.evaluate(data_samsum_pt['test'])
```

# Model Saving

```
from google.colab import drive from transformers
import PegasusForConditionalGeneration

drive.mount('/content/drive')
model_peg.save_pretrained('/content/drive/MyDrive/pretrained_data')
```

Chapter 9
Deployment

CHAPTER-9
DEPLOYMENT

In this chapter, we outline our deployment plans for the "Design and Development of Abstractive Text Summarization." The model is currently in development, and while it has not been deployed yet, we provide an overview of our deployment strategy.

9.1 Hosting Environment

We are in the process of selecting an appropriate hosting environment based on factors such as scalability, performance, and security.

9.2 Model Deployment Strategy (Planned)

Once the model is ready, we will serialize it into a deployable format, integrate it into a web application, and enable text summarization.

9.3 User Accessibility (Planned)

We plan to make the web-based interface accessible to users through a user-friendly URL.

9.4 Scalability (Planned)

Our deployment strategy will consider scalability to accommodate an increasing number of users and larger datasets.

9.5 Monitoring and Maintenance (Planned)

A monitoring and maintenance plan will be established to ensure the model's continued performance.

The deployment of the abstractive text summarization model is currently in the planning stage, and the steps and considerations mentioned in this chapter will be implemented when the model is ready for deployment.

APPENDIX-1 GLOSSARY OF TERMS

(In alphabetical order)

B	
Bias in Data	Unintended patterns in the data that can affect the model's predictions, often requiring mitigation during preprocessing.
D	
Data Collection	The process of gathering relevant data from various sources to be used for model training.
Data Preprocessing	Cleaning and preparing data to ensure its quality and suitability for model training.
Deployment	The process of making the model accessible to users for real-time predictions.
F	
Feature Engineering	The process of selecting and creating relevant features from the dataset to improve the model's predictive accuracy.
Feature Extraction	The process of selecting and creating relevant features from the dataset to improve the model's predictive accuracy.
G	
Geographical Variations	Differences in car prices due to location or region, which need to be considered in the model.
H	
Hosting Environment	The infrastructure or platform where the application and model will be hosted and made accessible to users.
M	
Machine Learning Model	A mathematical model that uses algorithms to predict or classify data based on input features. In this project, it predicts car prices based on car attributes.
Model Evaluation	The process of assessing the performance and accuracy of the machine learning model using various metrics.

Monitoring and Maintenance	A plan for continuous monitoring and upkeep of the model and application to ensure optimal performance.
S	
Scalability	The system's ability to handle an increasing number of users and larger datasets without a significant decrease in performance.
Serializing	Converting the machine learning model into a deployable format, often used for saving and loading models.
U	
User Interface	The graphical or web-based interface through which users can interact with the car price prediction model.
V	
Version Control	The practice of tracking changes made to code and other project files, often managed using tools like Git.

REFERENCES

Tokenization: Traditionally, techniques like tokenization (provided by libraries like NLTK [1]) were used for text processing and summarization. (NLTK Website: [https://www.nltk.org/] (https://www.nltk.org/))

Transformers: However, transformers [3] (Vaswani et al., 2017) revolutionized the field due to their ability to capture long-range dependencies in text. (Vaswani et al., 2017: [https://arxiv.org/abs/1706.03762] (https://arxiv.org/abs/1706.03762))

NLTK: The Natural Language Toolkit (NLTK) [1] (https://www.nltk.org/) provides a comprehensive suite of libraries and tools for various NLP tasks.

Long Short-Term Memory (LSTM): LSTMs [2] (https://en.wikipedia.org/wiki/Long_short-term_memory) are a type of RNN architecture known for handling long-term dependencies in data.

Transformers Library: The Transformers library [4] provides pre-trained models and functionalities for various NLP tasks, including text summarization. This code leverages the AutoModelForSeq2SeqLM class to load the pre-trained PEGASUS model [6] (Zhang et al., 2019). (Hugging Face Transformers: [https://huggingface.co/docs/transformers/index] (https://huggingface.co/docs/transformers/index))

Datasets Library: The Datasets library [5] simplifies loading and processing large datasets, while DataCollatorForSeq2Seq and TrainingArguments classes from Transformers [4] prepare data for training and define hyperparameters, respectively. The core training process is handled by the Trainer class, which utilizes the model, arguments, tokenizer, data collator, and training/evaluation datasets. (Hugging Face Datasets: [https://huggingface.co/docs/datasets/en/index] (https://huggingface.co/docs/datasets/en/index))

PEGASUS Model: Pre-trained models like PEGASUS are particularly well-suited for text summarization due to their ability to handle both encoder and decoder functionalities within the transformer architecture. However, other options like BART [7] (Lewis et al., 2019), T5 [8] (Raffel et al., 2020), and BERT [9] (Devlin et al., 2018) can also be explored depending on specific needs and resources. (Zhang et al., 2019: [invalid URL removed])

Transfer Learning: Fine-tuning the pre-trained model on a specific summarization dataset (like CNN/Daily Mail) leverages transfer learning [10] (Pan & Yang, 2009) to adapt the model's knowledge to the task. The training loop iterates through batches of data, updates model weights, and evaluates the model periodically. (Pan & Yang, 2009: [invalid URL removed])

Evaluation Metrics: While ROUGE score [11] (Lin, 2004) is a common metric, exploring additional evaluation methods like human evaluation or BLEU score can provide a more comprehensive understanding of summarization quality. (Lin, 2004: [https://www.aclweb.org/anthology/P04-1074] (https://www.aclweb.org/anthology/P04-1074))

Flask Framework: Finally, the trained model and tokenizer can be saved for future use, and frameworks like Flask [12] can be used to deploy the model as a web application for real-time text summarization. (Flask documentation: [https://palletsprojects.com/p/flask/] (https://palletsprojects.com/p/flask/))

PROJECT SUMMARY

About Project

Title of the project	Design & Development of Abstractive Text Summarization
Semester	6th
Members	3
Team Leader	Sanskar Agrawal
Describe role of every member in the project	Sanskar Agrawal: He helped with the model development and back-end connectivity. Shivam pathak: He helped with Web development and connectivity. Aryan Sharma: He helped with Model development and web design.
What is the motivation for selecting this project?	In today's information age, we drown in text. This project cuts through the noise with an automatic text summarization system, empowering users to grasp key points of lengthy documents faster. It fosters a more informed and productive society by saving time and effort for researchers, students, and professionals, ultimately leading to better decision-making and a brighter future empowered by knowledge.
Project Type **(Desktop Application, Web Application, Mobile App, Web)**	Web Application

Tools &Technologies

Programming language used	Python
Interpreter used **(with version)**	Python (above 3)
IDE used **(with version)**	Visual Studio Code (version 1.84)

Front End Technologies **(with version, wherever Applicable)**	HTML 5, CSS, Django
Back End Technologies **(with version, wherever applicable)**	• Python 3.11.4 • Flask==3.0.2
Database used **(with version)**	-

Software Design & Coding

Is prototype of the softwaredeveloped?	Yes
SDLC model followed (Waterfall, Agile, Spiral etc.)	-
Why above SDLC model isfollowed?	-

Justify that the SDLC model mentioned above is followed in the project.	-
Software Design approach followed **(Functional or Object Oriented)**	-
Name the diagrams developed **(according to the Design approach followed)**	-
In case Object Oriented approach is followed, which of the OOPS principles are covered in design?	-
No. of Tiers **(example 3-tier)**	-
Total no. of frontend pages	3
Total no. of tables in database	-
Database is in which Normal Form?	-
Are the entries in database encrypted?	No
Front end validations applied **(Yes / No)**	Yes
Session management done **(in case of web applications)**	No
Is application browser compatible **(in case of web applications)**	Yes
Exception handling done **(Yes / No)**	Yes
Commenting done in code **(Yes / No)**	Yes
Naming convention followed **(Yes / No)**	Yes
What difficulties faced during deployment of project?	Problems we faced were: 1. Finding the accurate dataset 2. Getting the required accuracy

	3. Linking it with the Flask
Total no. of Use-cases	
Give titles of Use-cases	-

Project Requirements

MVC architecture followed **(Yes / No)**	**Yes**
If yes, write the name of MVC architecture followed **(MVC-1, MVC-2)**	**MVC-2**
Design Pattern used **(Yes / No)**	**Yes**
If yes, write the name ofDesign Pattern used	**MVT (Model View Template)**
Interface type **(CLI / GUI)**	**GUI**
No. of Actors	**-**
Name of Actors	**-**
Total no. of Functional Requirements	
List few important non- Functional Requirements	

Testing

Which testing is performed? **(Manual or Automation)**	Manual

Is Beta testing done for this project?	No

Write project narrative covering above mentioned points

Project Summary:

- **Title:** Design & Development of Abstractive Text Summarization Model
- **Semester:** 6th
- **Members:** 3
- **Team Leader:** Sanskar Agrawal

Roles:

- Sanskar Agrawal: He helped with the model development and back-end connectivity.
- Shivam Pathak: He helped with web development.
- Aryan Sharma: He helped with model development and web design.

Motivation: To summarize large text and provide valuable summary.

Project Type: Web Application

Technologies:

- Programming Languages: Python (version 3 and above)
- IDE: Visual Studio Code (version 1.84), Google Colab
- Front End: HTML 5, CSS
- Back End: Flask, Python 3.11.4

Software Development:

- Prototype developed.
- Agile SDLC model followed.
- Object-Oriented design approach.

Project Details:

- Web app with - front-end pages,
- Front-end validations,browser compatibility, exception handling, codecommenting, and naming conventions are implemented.

Challenges During Deployment:

1. Finding an accurate dataset.
2. Achieving desired prediction accuracy.
3. Linking the model with Flask.

Requirements:

- Follows MVC-2 architecture.
- Design pattern used: MVT (Model-View-Template).
- Interface: GUI

Testing:

- Manual testing.
- No beta testing conducted.

Sanskar Agrawal	Shivam Pathak	Aryan Sharma	Guide Signature
0187AD211036	0187AD211037	0187AD211010	(Project Guide name)