

**Department of Electrical and Computer Engineering**  
**North South University**

**Senior Design Project**

**SummaryScribe: Transformative Meeting Summarization using Prompt Engineering with Retrieval Augmented Generation (RAG) Technique Based on Natural Language Processing**

<b>Sheikh Mohammed Wali Ullah</b>	<b>2021186042</b>	<b>Section - 10</b>
<b>Zobaer Ahammod Zamil</b>	<b>2021796042</b>	<b>Section - 10</b>
<b>Md Saiyem Raiyan</b>	<b>2012468042</b>	<b>Section - 10</b>
<b>Samia Sultana</b>	<b>2014048042</b>	<b>Section - 23</b>

**Faculty Advisor:**

**Dr. Mohammad Ashrafuzzaman Khan**

**Assistant Professor**

**Department of Electrical and Computer Engineering**

**Spring, 2024**

## DECLARATION

This is to declare that this project is our original work. No part of this work has been submitted elsewhere, partially or fully, for the award of any other degree or diploma. All project-related information will remain confidential and shall not be disclosed without the formal consent of the project supervisor. Relevant previous works presented in this report have been properly acknowledged and cited. The plagiarism policy, as stated by the supervisor, has been maintained.

Students' names and signatures

-----  
**1. Sheikh Mohammed Wali Ullah (ID # 2021186042)**

-----  
**2. Zobaer Ahammod Zamil (ID # 2021796042)**

-----  
**3. Md Sayem Raiyan (ID # 2012468042)**

-----  
**4. Samia Sultana (ID # 2014048042)**

## ACKNOWLEDGEMENTS

The authors would like to express their heartfelt gratitude towards their project and research supervisor, Dr. Mohammad Ashrafuzzaman Khan, Assistant Professor, Department of Electrical and Computer Engineering, North South University, Bangladesh, for his invaluable support, precise guidance, and advice pertaining to the experiments, research and theoretical studies carried out during the course of the current project and also in the preparation of the current report.

Furthermore, the authors would like to thank the Department of Electrical and Computer Engineering, North South University, Bangladesh, for facilitating the research. We would also like to thank my friends Sheikh Mohammed Wali Ullah, Zobaer Ahammod Zamil, Md Saiyem Raiyan, and Samia Sultana for helping us with this project. The authors would also like to thank their loved ones for their countless sacrifices and continual support.

## ABSTRACT

### **SummaryScribe: Transformative Meeting Summarization using Prompt Engineering with Retrieval Augmented Generation (RAG) Technique Based on Natural Language Processing**

The model aims to generate a better contextual summary from news or long transcripts using a SLM model. It involves a thorough process of model evaluation and optimization. Initially, multiple models, including Pegasus-CNN-DailyMail, EleutherAI/gpt-neo-2.7B, EleutherAI/gpt-neo-125M, were analyzed to determine their effectiveness in summary generation. Among these, we found a better model - the Microsoft Phi-2 model, a small language model (SLM), as the best performer. To enhance the quality of the generated summaries, we utilized various prompt engineering techniques such as zero-shot, few-shot, and Retrieval-Augmented Generation (RAG). The implementation of RAG (Retrieval-Augmented Generation) makes the summaries much better by adding more context and retrieving useful information. Furthermore, using low-cost fine-tuning and prompt engineering techniques, this model can preserve the semantics and meaning of the textual context properly.

## TABLE OF CONTENTS

DECLARATION.....	2
ACKNOWLEDGEMENTS.....	3
ABSTRACT.....	4
TABLE OF CONTENTS.....	5
LIST OF FIGURES.....	6
Chapter 1 Introduction.....	7
1.1    Background and Motivation.....	7
1.2    Purpose and Goal of the Project.....	9
1.3    Organization of the Report.....	11
Chapter 2 Research Literature Review.....	12
2.1 Existing Research and Limitations.....	12
Chapter 3 Methodology.....	15
3.1 System Design.....	15
3.2 Software Components.....	20
3.3 Software Implementation.....	21
Chapter 4 Investigation/Experiment, Result, Analysis and Discussion.....	28
Chapter 5 Impacts of the Project.....	40
5.1 Impact of this project on societal, health, safety, legal and cultural issues.....	40
5.2 Impact of this project on environment and sustainability.....	41
Chapter 6 Project Planning .....	45
Chapter 7 Conclusions.....	46
7.1 Summary.....	46
7.2 Limitations.....	46
7.3 Future Improvement.....	47
References.....	49

## LIST OF FIGURES

<b>Figure 1</b> Summary	<b>10</b>
<b>Figure 2</b> System Design of the model using Pegasus-CNN-Dailymail model	<b>15</b>
<b>Figure 3</b> Design of the solution of problem	<b>16</b>
<b>Figure 4</b> EleutherAI/gpt-neo 2.7B and 125M with Happy Transformer	<b>16</b>
<b>Figure 5</b> Architecture of Phi-2	<b>17</b>
<b>Figure 6</b> Architecture of Phi-2 with Prompt Engineering	<b>18</b>
<b>Figure 7</b> Architecture of Zephyr	<b>19</b>
<b>Figure 8</b> Code snippets using pipe	<b>21</b>
<b>Figure 9</b> Code snippets of training arguments	<b>21</b>
<b>Figure 10</b> Code snippets of trainer	<b>21</b>
<b>Figure 11</b> Dataset Summary	<b>22</b>
<b>Figure 12</b> Happy Transformer Loading	<b>23</b>
<b>Figure 13</b> Happy Transformer Loading	<b>23</b>
<b>Figure 14</b> Phi-2 Loading	<b>24</b>
<b>Figure 15</b> Phi-2 Loading Tokenizer	<b>25</b>
<b>Figure 16</b> Phi-2 Output Code	<b>25</b>
<b>Figure 17</b> RAG implement (all 3)	<b>26</b>
<b>Figure 18</b> Zephyr implement and output	<b>27</b>
<b>Figure 19</b> Code snippets of generating summary from the samsum dataset before train	<b>28</b>
<b>Figure 20</b> Code snippets of generating summary from the samsum dataset after train	<b>30</b>
<b>Figure 21</b> Generated summary comparison with ChatGPT summary	<b>30</b>
<b>Figure 22</b> ROUGE score before training and graph	<b>32</b>
<b>Figure 23</b> ROUGE score after training and graph	<b>32</b>
<b>Figure 24</b> ElutherAI/gpt-neo training results	<b>33</b>
<b>Figure 25</b> phi-2 results with prompt engineering techniques	<b>34</b>
<b>Figure 26</b> phi-2 with RAG results	<b>35</b>
<b>Figure 27</b> Zephyr results	<b>36</b>
<b>Figure 28</b> Rouge and Bleu score	<b>36</b>
<b>Figure 29</b> OpenAI vs Phi-2 comparison with hemingway vs grammarly score	<b>37</b>
<b>Figure 30</b> Hemingway score	<b>37</b>
<b>Figure 31</b> Grammarly score	<b>38</b>
<b>Figure 32</b> Hemingway and Grammarly score for 10 samples	<b>39</b>
<b>Figure 33</b> All models (methodology) Roadmap	<b>40</b>
<b>Figure 34</b> Project Planning Gantt chart	<b>46</b>

# Chapter 1 Introduction

## 1.1 Background and Motivation

[21] In today's fast-paced business environment, effective communication and documentation of meetings are essential for collaboration and decision-making. However, manually summarizing discussions can be time-consuming and prone to errors. To address this challenge, our project, Autonote, aims to leverage Natural Language Processing (NLP) techniques to automate the summarization of meetings. The goal of NLP meeting summarization is to make it easier for people to understand and use the information generated in meetings. This system can lead to several benefits, such as improved communication, collaboration, decision-making, and productivity.

### Summarization

Summarization is the process of condensing a longer piece of text, such as an article, book, or document, into a shorter version while retaining its essential information, main points, and critical ideas. It is a valuable skill for students and professionals, allowing for quick overviews, key points identification, comparisons, summary creation, and report generation. It saves time, enhances learning, and improves communication.

It can be extractive, abstractive, single-document, multi-document, generic, query-based, update, or opinion summarization. Extractive summarization uses algorithms to select and combine sentences from the source text, while abstractive summarization uses natural language generation techniques. Generic summarization is broadly applicable and informative, while query-based summarization generates a summary responding to a specific query or question. Update summarization summarizes new information concerning existing knowledge. Text summarization is a Natural Language Processing (NLP) process that uses Deep Learning and Machine Learning models to break down significant texts into their most essential parts, either for static or audio/video streams.

In the context of our project, "**SummaryScribe**," summarization refers to the automated process of condensing the content of meeting or news into shorter, more digestible versions while retaining the essential information, key discussion points, action items, and decisions. This feature of SummaryScribe streamlines the review and reference of meeting content, making it easier for participants to grasp the main ideas without the need to sift through the entire meeting or news transcript.

### Natural Language Processing (NLP)

[21][23]Natural language processing (NLP) refers to the branch of computer science and, more specifically, the branch of artificial intelligence or AI concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its whole meaning, complete with the speaker or writer's intent and sentiment.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it is ingesting. Some of these tasks include the following:

- **Speech recognition:** It is also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions.

- **Part of speech tagging:** It is also called grammatical tagging, which is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies ‘make’ as a verb in ‘I can make a paper plane’ and as a noun in ‘What make of car do you own?’
- **Word sense disambiguation:** It is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determines the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).
- **Coreference resolution:** Identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., ‘she’ = ‘Mary’), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which ‘bear’ is not an animal but a large hairy person).
- **Sentiment analysis:** It attempts to extract subjective qualities, attitudes, emotions, sarcasm, confusion, and suspicion from the text.
- **Natural language generation:** It is sometimes described as the opposite of speech recognition or speech-to-text; it is the task of putting structured information into human language.

## NLP work

[23] Computational linguistics involves understanding and constructing human language models using computers and software. Machine learning trains computers to understand human language features like sarcasm, metaphors, and grammar. Deep learning, a specific field of machine learning, teaches computers to learn and think like humans. NLP implementation involves gathering and preparing unstructured text or speech data, pre-processing using tokenization, stemming, lemmatization, and stop word removal, training models with large data samples, and deploying and inferring the model's output. This process allows for the application to be run on live data and obtain the required output.

## Approaches to Natural Language Processing

[20] Natural language processing (NLP) approaches include supervised and unsupervised, natural language understanding, and natural language generation. Supervised NLP trains software with labeled input and output, while unsupervised NLP uses a statistical model to predict patterns in non-labeled input. Natural language understanding analyzes sentence meanings, while natural language generation produces conversational text based on specific keywords or topics. These approaches help software categorize documents, predict sentence patterns, and produce conversational text.

## 1.2 Purpose and Goal of the Project

### Problem Clarification

In modern workplaces, meetings are essential for collaboration and decision-making. Automatic meeting summarization, powered by Natural Language Processing (NLP), aims to address this challenge by distilling key meeting insights into concise summaries. Sometimes, we miss meetings that are happening online due to some unavoidable circumstances. Again, sometimes, we miss out on essential meeting points for various reasons. However, before our important work or study time is very crucial, we need to do it in a faster or optimal way. That is why we need to summarize a long meeting so that we can understand quickly and quickly within our due time. Therefore, the gist of an online meeting is often very important to those who missed it or tried to recall it. There can be a system that can make a summary of a meeting while it is running online.

The focus point of this system is to generate better contextual summaries from news or long transcripts. After taking a news text or long transcript, we feed that into the fine tuned model to generate a summary.

#### I. What is NLP doing?

This NLP summarizing develops a system that can automatically generate accurate and informative summaries of long meeting transcripts.

#### II. Generated Summary

Introducing a revolutionary concept: a dynamic system that crafts comprehensive summaries of online meetings as they unfold. The generated summary should capture the essence of the long transcript while filtering out redundant or less relevant information.

#### III. System Ability

The Autonote project seeks to automate the summarization of meetings to:

1. Provide participants with concise and coherent summaries that capture the main discussion points, key decisions, and action items.
2. Reduce the time and effort required for manual note-taking during meetings.
3. Enhance the tracking and management of action items, improving accountability and task follow-through.
4. Facilitate knowledge sharing and decision-making by making meeting insights easily accessible.

By addressing these problem statements, SummaryScribe aims to revolutionize the way organizations conduct and benefit from meetings, ultimately leading to increased efficiency, better collaboration, and more informed decision-making.

### Novelty

Plenty of research on summarization exists, but by analyzing Hemingway grade, ROUGE, and BLEU scores, we demonstrate superior performance compared to existing tools. Through rigorous analysis of multiple models, we identify the most effective ones for generating concise, coherent, and relevant meeting summaries by using Microsoft Phi-2 and RAG.

## LLM and SLM Ability

Large Language Models (LLMs) like GPT-4 are advanced AI models trained on vast amounts of text data, capable of tasks such as writing, translation, and complex conversation. They require significant computational resources. In contrast, Small Language Models (SLMs) are more limited in scope and size, suited for specific tasks like simple chatbots, text classification, and sentiment analysis, requiring less computational power. LLMs offer broader capabilities and higher accuracy, while SLMs are more efficient for targeted applications. LLMs provide detailed, nuanced summaries with high contextual understanding, ideal for complex summarization tasks. SLMs offer simpler, more straightforward summaries, efficient for basic summarization needs. By using prompt engineering, we can enhance both LLMs and SLMs to generate more accurate and relevant summaries tailored to our requirements.

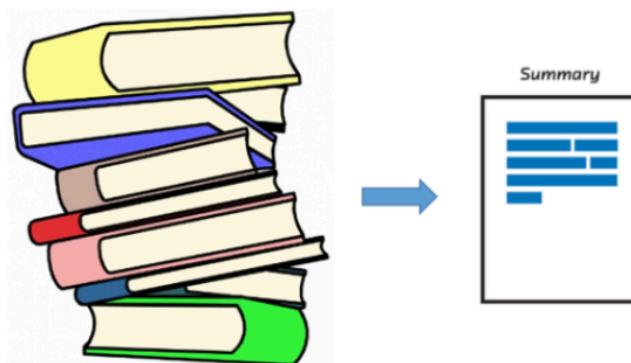
## Summarization with NLP

[19][20]Summarization with Natural Language Processing (NLP) refers to using NLP techniques and algorithms to automatically generate concise and coherent summaries of longer texts, documents, or spoken content. NLP-based summarization methods can be broadly categorized into two main approaches:

NLP-based summarization has a wide range of applications, for example:

- **News Summarization:** Automatically generating summaries of news articles to provide readers with the main points.
- **Document Summarization:** Condensing lengthy reports, research papers, or legal documents for quick reference.
- **Meeting Summarization:** As in your Autonote project, summarize the content of meetings to capture key discussions and action items.
- **Content Curation:** Creating concise descriptions for content aggregation, such as blog posts or product descriptions.
- **Search Engine Snippets:** Generating brief descriptions for search engine results to help users quickly assess the relevance of web pages.

So, our models also summarize the Large Language Model (LLM). The focus point of this system is to generate better contextual summaries from news or long transcripts. After taking a news text or long transcript, we feed that into the fine tuned model to generate a summary. However, analyzing multiple models pegasus-cnn-dailymail, EleutherAI/gpt-neo 125M, EleutherAI/gpt-neo-2.7B, we found a better model, Microsoft Phi-2, which is a LLM model. We used the model with prompt engineering techniques like zero-shot, few-shot and Retrieval Augmented Generation. Using RAG on the models improves the generated summary significantly. Moreover, if we tune the parameters like temperature and num\_of\_tokens, we usually get even better summaries.



*Figure 1: Summary*

## 1.3 Organization of the Report

No.	Topic	Describe
1	Introduction	About project, Summarization, Natural language Processing, It works, approaches Purpose and goals like problem clarification, summarization correlated with NLP
2	Research Literature Review	What we research before and after the project starting, Read almost 25/30 papers and pick ten important papers and reviews it with authors name mentioned and limitations
3	Methodology	How we designed our system, hardware/software components and implemented them. How we use our models and correlation with datasets and how to solve issues regarding this.
4	Investigation/Experiment, Result, Analysis and Discussion	Project-related investigation, dataset analysis, comparison between the model with ChatGPT and Hugging face API, Context missing in case of the long transcript, 12 field testing that exists in our model, New model exploration, new model limitations, brush up new model
5	Impacts of the Project	How our project impacts on societal, health, safety, legal, cultural issues, environment, sustainability, usability and manufacturing
6	Project Planning and Budget	If we think about a business project and real-world implementation what will be the budget and planning
7	Conclusion	So far what we did, what the project says, its limitations and finally how it beneficial in the future modern world with a long run

# Chapter 2 Research Literature Review

## 2.1 Existing Research and Limitations

With the advent of deep learning, abstractive summarization techniques emerged, allowing models to generate summaries that are not just extracts but paraphrases of the original content. These methods have shown significant improvements in producing coherent and fluent summaries.

A comprehensive review by **Saiyyad and Patil (2024)**<sup>[1]</sup>, “Text Summarization Using Deep Learning Techniques: A Review” [1], highlights various deep learning techniques, including neural networks and transformers, which have revolutionized text summarization by enabling models to understand and generate human-like text.

**Hanlei Jin, Yang Zhang (2024)**<sup>[2]</sup> surveyed adopts a process-oriented schema to categorize text summarization methods, focusing on practical applications in real-world scenarios. It also explores the integration of Large Language Models (LLMs) in the automatic text summarization process.

**Kumar, S., Solanki, A. (2023)** [3], proposed an ATS model using a Transformer Technique with Self-Attention Mechanism (T2SAM) that model improves the performance of text summarization and is trained on the Inshorts News dataset combined with the DUC-2004 shared tasks dataset. The performance of the proposed model has been evaluated using the ROUGE metrics, and it has been shown to outperform the existing state-of-the-art baseline models. This paper presents a novel approach to abstractive text summarization using transformer models, particularly focusing on their application to lengthy documents. The study demonstrates how transformer-based models can generate high-quality summaries by understanding and rewriting the content.

**Basyal, L., & Sanghvi, M. (2023)** [4], as shown remarkable promise in enhancing summarization techniques. Their paper embarks on an exploration of text summarization with a diverse set of LLMs, including MPT-7b-instruct, falcon-7b-instruct, and OpenAI ChatGPT text-davinci-003 models. The experiment was performed with different hyperparameters and evaluated the generated summaries using widely accepted metrics such as the BLEU Score, ROUGE Score, and BERT Score. According to the experiment, text-davinci-003 outperformed the others. This investigation involved two distinct datasets: CNN Daily Mail and XSum. Its primary objective was to provide a comprehensive understanding of the performance of LLMs when applied to different datasets.

**Zhang, T., Ladhak, F., Durmus, E., Liang, P., McKeown, K., & Hashimoto, T. B. (2023)** [5], benchmarked several large language models, including different versions of GPT-3 and other models like Pegasus and BRIO, for their performance in news summarization. It evaluates the models using human ratings on criteria such as faithfulness, coherence, and relevance.

**Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024)** [6] provide a comprehensive overview of recent advancements in prompt engineering, categorized by application area. The paper details various prompting methodologies, their applications, the models involved, and the datasets used. It discusses the strengths and limitations of each approach and includes a taxonomy diagram and a table summarizing key points, models, and datasets for each technique. The survey categorizes and reviews techniques such as zero-shot, few-shot, and chain-of-thought prompting, and assesses their effectiveness in enhancing LLM performance across different tasks. This structured analysis aims to aid in understanding the field's rapid development and to highlight open challenges and future opportunities in prompt engineering.

**Jingqing Zhang et al [15]** developed a model named **PEGASUS**: Pre-training with Extracted Gap-sentences for Abstractive Summarization, that proposes pre-training large Transformer-based encoder-decoder models on massive text corpora with a new self-supervised objective. In

PEGASUS, important sentences are removed/masked from an input document and are generated together as one output sequence from the remaining sentences, similar to an extractive summary. It summarizes shorter texts but struggles with longer transcripts, resulting in unsuitable summaries and leading to a loss of context.

**Ming Zhong et al. [7]** developed a **DIALOGLM**, a Pre-trained Model for Long Dialogue Understanding and Summarization model that converts a long text into a summary case of a conversation. Using NLP and the concept of summarization, they are converting a speech which is a long speech into a summary. DIALOGLM, a pre-trained neural encoder-decoder model for long dialogue understanding and summarization. The benefits of this model are more pronounced in datasets with longer input text. With further pre-training, the model deals with long conversation scenarios. The results demonstrate that each dialogue-inspired noise and the proposed hybrid attention methods can further improve the model.

**Ravali Boorugu and Dr. G. Ramesh [8]** presented an NLP-based approach to **summarize product reviews**, employing different summarization methods for accuracy. Niladri Chatterjee, Amol Mittal, and Shubham Goyal introduced an extractive summarization technique using Genetic Algorithms to condense information from multiple documents, aiming to save time and enhance efficiency. Amol Tandel et al. proposed a multi-document summarization technique utilizing LexRank to select relevant data. Aditya Jain et al. focused on salient sentence recognition and organizing information in an extractive model. Others explored problems with word embedding, solved by Mikolov et al., while Yang et al. proposed a query-based summarization model using search engines for document context.

**Gabriel Murray and Giuseppe Carenini [9]** developed a **Summarizing Spoken and Written Conversations** model that summarizes conversations in meetings and emails using a system that uses general conversational features across multiple domains, comparing its results with domain-dependent systems. This research takes an extractive approach to summarization, presenting conversational features for locating the most salient sentences in meeting speeches and emails.

**Yashar Mehdad Giuseppe Carenini Frank W. Tompa Raymond T. NG [10]** developed an **Abstractive Meeting Summarization with Entailment and Fusion** and proposed a framework for abstractive meeting summarization, clustering sentences into communities and building an entailment graph to identify relevant sentences. This work presents a comprehensive pipeline for producing abstractive summaries for each meeting transcript. Word graph-based method to aggregate and generate abstractive sentence summaries, extending Filippova's method. Instead of utilizing the traditional NLG pipeline and a word graph technique to capture the semantics of the meeting, our objective is to construct a meeting summary using a subset of annotations, including a human synopsis and connections to source meeting sentences. The CONGA algorithm is used to identify which nodes can be clustered as a community to generate an abstract sentence, using betweenness scores to detect communities in a graph.

**Hamza Shabbir Moiyadi1 et al [11]** developed **NLP Based Text Summarization Using Semantic Analysis**. A technique for generating the summarization of domain-specific text by using Semantic Analysis for text summarization. The input document is first parsed or pre-processed, wherein there is a removal of unneeded words such as ‘stop words,’ which are simply small function words, like “the,” “and” “a,” which do not contribute meaning to the text summary. In the summarization process, the system arranges the sentences generated from the SVD Analysis stage by semantically placing them so that the summary encompasses all the concepts of the original text.

**Chinatsu Aone, Mary Ellen Okurowski, and James Gorlinsky [12]** developed a model called **Trainable, Scalable Summarization Using Robust NLP and Machine Learning**, which uses features from retrieval, extraction, NLP techniques and online resources. It combines these features using a machine learning algorithm, demonstrating system scalability and usability. Trainable and

scalable summarization system that utilizes features derived from information retrieval, information extraction, and NLP techniques and online resources.

**Yash Agrawal, Atul Thakre, Tejas Tapas, Ayush Kedia, Yash Telkhade, and Vasundhara Rathod** [13] developed a model, **Comparative Analysis of NLP models for Google Meet Transcript Summarization**, the paper illustrates the comparison of 3 models, which is used to summarize google meetings. After removing articles, prepositions, and other unnecessary parts, they used tokenization and lemmatization. Lastly, they used the NLTK (Natural Language Toolkit) model, Word Embedding Model, and T5 model. Moreover, perform ROUGE analysis. Two models use extractive summarization, and one model for abstractive summarization. These models used frequency-based, score-based, matrix-based, and mapping-based technologies. Also used pre-trained datasets such as Glove. After training those models, using ROUGE analysis, they checked which one works more accurately without changing the meaning of the context.

**Neslihan Akar and Metin Turan** [14] developed a model, **A General Approach for Meeting Summarization: From Speech To Extractive Summarization**, which proposes a model to summarize a meeting using Natural Language Processing text summarization techniques (Extractive Summarization) supported by a particular dictionary usage. After extractive summarize and human summaries, comparisons were interpreted by an average of all similarity rates. The first only uses the TF-IDF algorithm. The second is the dictionary form that does not use a “word ratio.” The third one used the dictionary form with 50% “word ratio”. The last one used the dictionary form with 20% “word ratio”. The automation of meeting summarization and extracting pivotal information holds immense potential to enhance operational efficiency across various business domains. The paper evaluates the summarization models by comparing the summaries generated by the model with those created by human summarizers. The alpha coefficient of 10.2 is used to optimize the model. The study also compares summaries generated by the model with human summaries. This focuses on extractive summarization, a technique where significant sentences are selected directly from the original content. In conclusion, this model finds application in diverse scenarios, such as condensing lengthy lectures, conferences, and meetings.

# Chapter 3 Methodology

## 3.1 System Design

### Google/pegasus-cnn\_dailymail

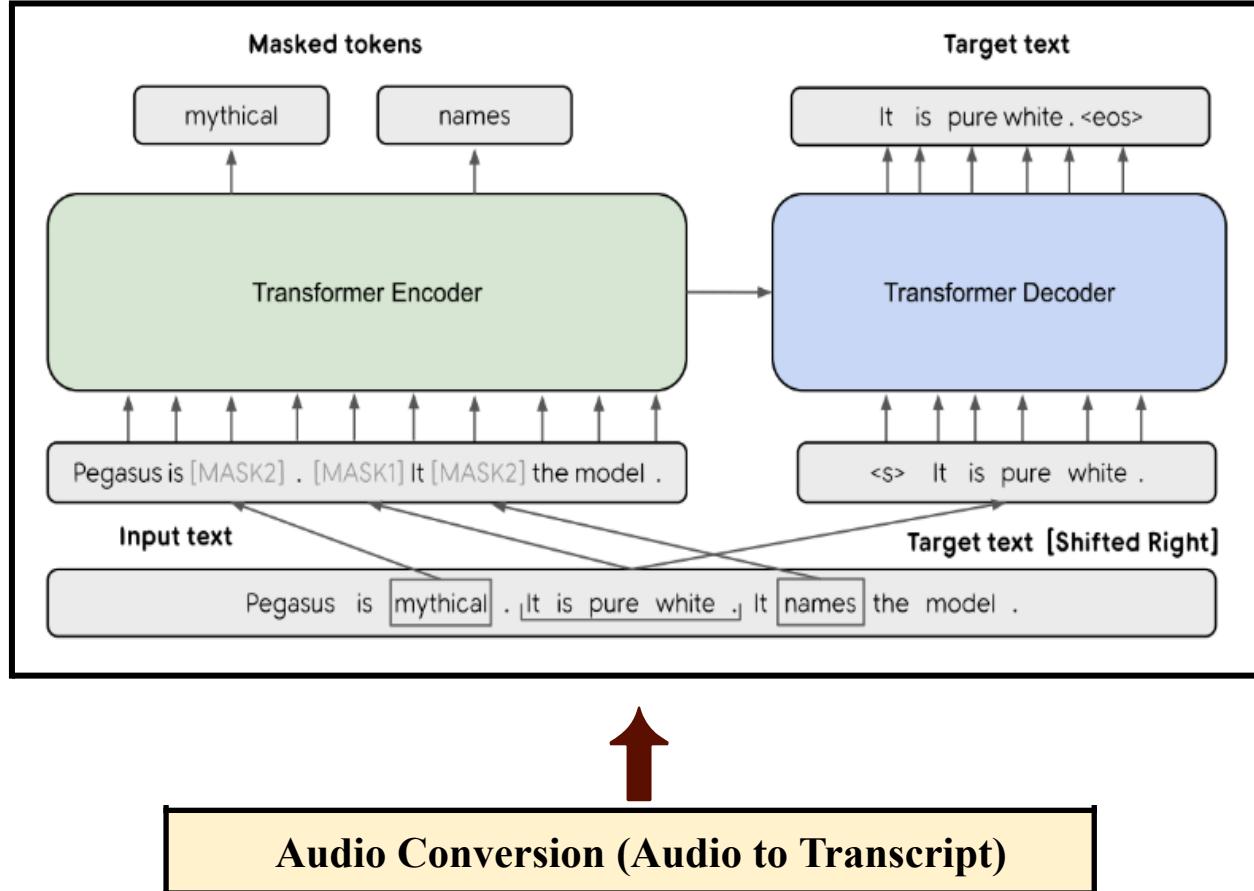


Figure 2: System Design of the model using Pegasus-CNN-Dailymail model

**Model Use:** google/pegasus-cnn\_dailymail

**Dataset Use:** samsum

The previous system generates a summary from the meeting transcript. **pegasus-cnn\_dailymail** model uses the “**samsum**” dataset where dialogue with summaries exists in that dataset. The hugging API generates the summary also from the transcript. The meeting audio generated a transcript conversion. Then, input the transcript and make the summary of the transcript. The data was collected from a YouTube meeting transcript from audio voice to caption of the subtitle. Therefore, it converted into audio to text. After that, segmentation is the part that divides the data into segments that can be easily analyzed, processed, and converted into text. We fine-tuned the previously selected **Pegasus-CNN-Dailymail** model.

According to these criteria, the model is fine-tuned and trained up to **15 epochs** and Tested with **12 fields** ( paper of Pegasus model) related data. So, in exploratory data analysis, we find a short summary with dataset validation. Besides, it handles problematic transcripts and irrelevant summaries. Our model type is a transformer that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence and evaluating the metrics in rouge scores. To get the best and optimal result, we fine-tuned the model with changing parameters and penalty scores and stratified it with a validation set. Here, we used Stochastic gradient descent (SGD) and normalization. Finally, evaluate the model with performance on validation and testing datasets.

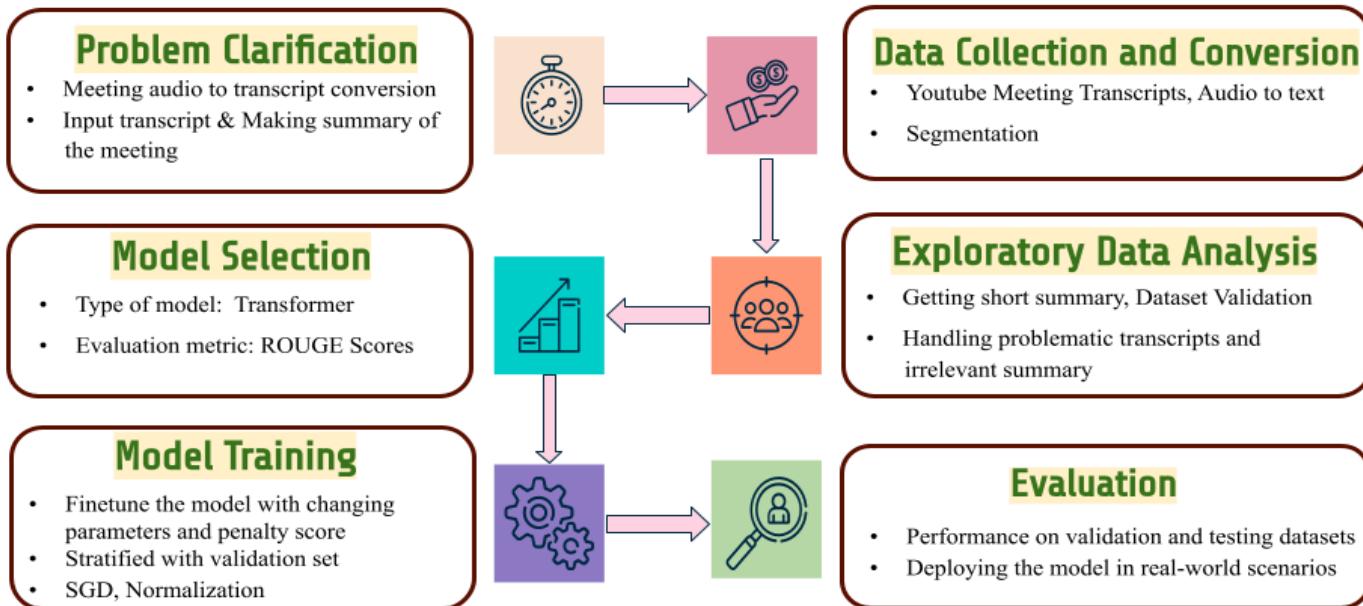


Figure 3: Design of the solution of problem

## EleutherAI/get-neo-2.7B + EleutherAI/get-neo-125M

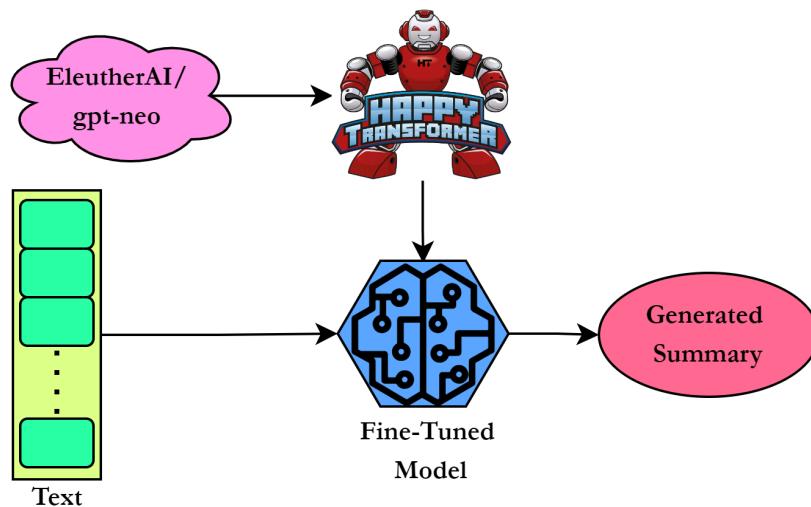


Figure 4: EleutherAI/gpt-neo 2.7B and 125M with Happy Transformer

**Model Use (Load): EleutherAI/gpt-neo-2.7B**

(<https://huggingface.co/EleutherAI/gpt-neo-2.7B>)

**Model Use (Load+Train): EleutherAI/gpt-neo-125M**

(<https://huggingface.co/EleutherAI/gpt-neo-125m>)

**Transformer Model: Happy Transformer** (<https://happytransformer.com/text-generation/>)

**Dataset Use: CNN-DailyMail News Text Summarization**

(<https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail>)

The system generates a summary from the meeting transcript. **EleutherAI/gpt-neo** model uses the “**CNN-DailyMail News Text Summarization**” dataset where dialogue with summaries exists in that dataset. Here, we load the model by a transformer model called “**Happy Transformer**”.

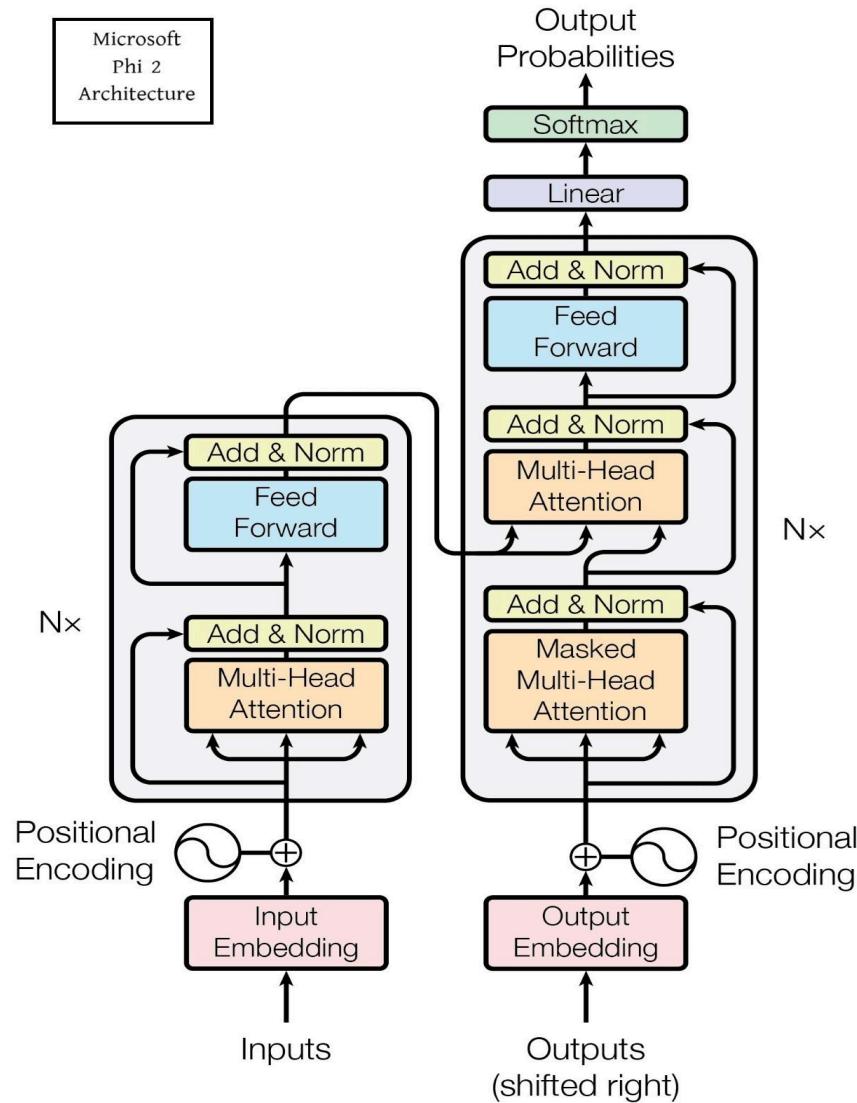
## Workings with Phi-2

**Shift with a new model** ➔ **microsoft/phi-2**

(<https://huggingface.co/microsoft/phi-2>)

**Dataset Use:** CNN-DailyMail News Text Summarization

(<https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail>)



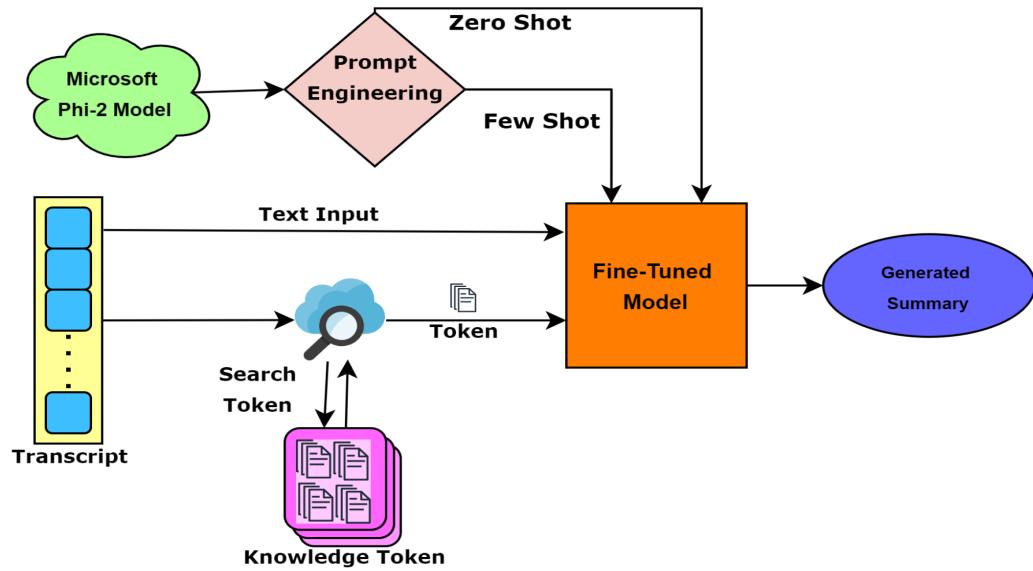
**Figure 5: Architecture of Phi-2**

The system generates a summary from the news transcript. “**Phi-2**” model uses the “**CNN-DailyMail News Text Summarization**” dataset where dialogue with summaries exists in that dataset. The meeting audio generated a transcript conversion. Then, input the transcript and make the summary of the transcript.

Most Importantly, To generate a more contextual and good summary we applied 3 major techniques of **Prompt Engineering**. The techniques are: Zero Shot, Few Shot and Retrieval-Augmented Generation (RAG). Basically, We use Microsoft Phi-2 for small language model (SLM) for better performing to generate more contextual and strong summary.

## Workings with Prompt Engineering Techniques with Phi-2

**Phi-2 by Prompt Engineering** – Using zero shot, few shot, and RAG technique.



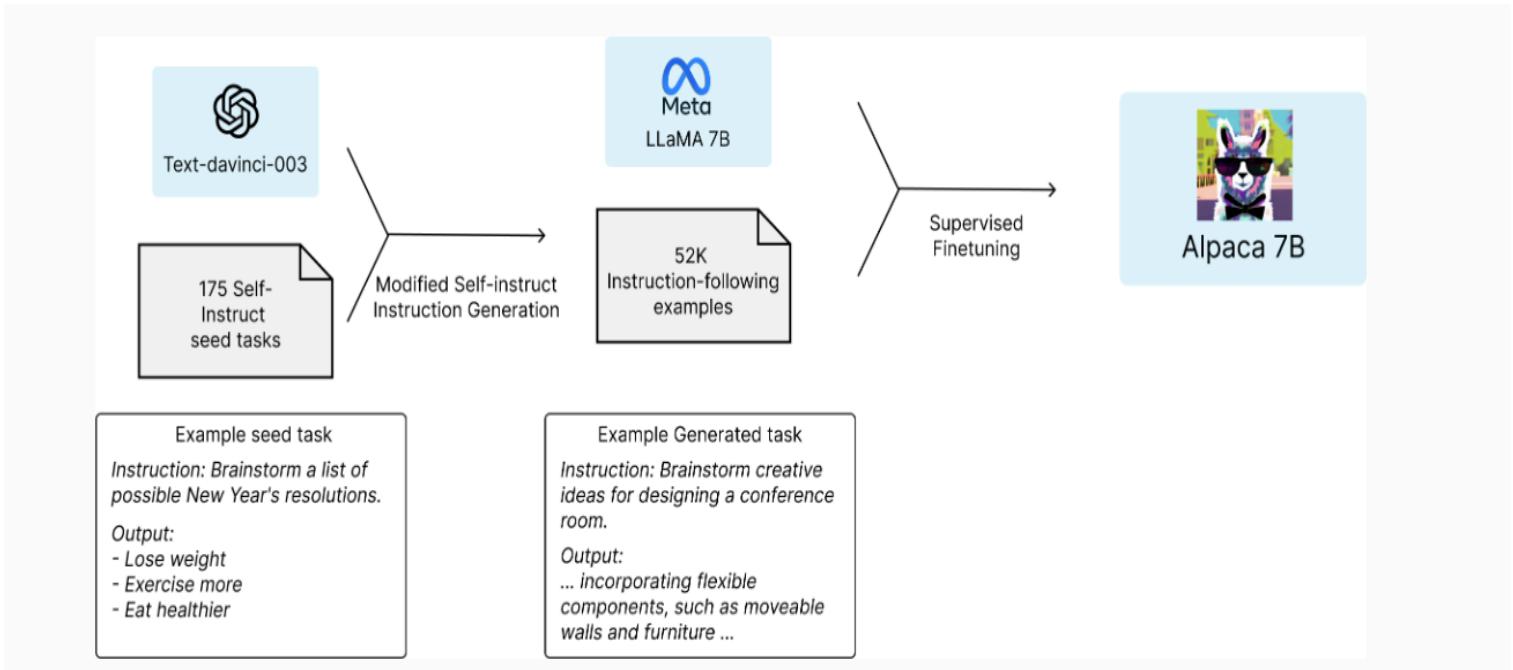
**Figure 6: Architecture of Phi-2 with Prompt Engineering**

**Working Process:** As mentioned in our title, we are utilizing Natural language Processing and implementing various models to generate the summary. After putting a text as input, the news transcript works through our models. We used several approaches. Among them, the Microsoft Phi-2 model is finetuned with zero shot, few shot, and RAG technique by prompt engineering, which works through our model and generates the summary of that particular transcript. Additionally, the model is perfectly analyzed according to the Rouge and Bleu score of the model.

**Zero-shot, Few-shot and RAG:** Zero-shot summarization uses a clear prompt to instruct the model to summarize a text, leveraging its general pre-trained knowledge without specific examples. Few-shot summarization provides the model with a few text-summary pairs as examples in the prompt, helping it recognize the summarization pattern and apply it to new texts more accurately. Finally, The Retrieval-Augmented Generation (RAG) technique for summarization combines retrieving relevant documents or passages from an external database with generating a summary, leveraging both the retrieval and generative capabilities to produce more accurate and contextually informed summaries.

In conclusion, Cross-match between the input transcript token and our knowledge-based token database, we gather the most similar token and pass it to our model along with the input transcript. Then, it generates a short contextual summary.

## Workings with Zephyr



**Figure 7: Architecture of Zephyr**

We tried to use HuggingFaceH4/zephyr-7b-beta with prompt engineering like microsoft/phi-2. So, we used kaggle notebook to load the HuggingFaceH4/zephyr-7b-beta model and tried to generate a summary from this. As it is a large model, our goal was to get a more contextual and semantically correct summary from this.

## 3.2 Software Components

List of Software/Hardware Tools

Tool	Functions	Other similar Tools (if any)	Why selected this tool
Kaggle	Coding implementation	Jupyter Notebook, colab	lack of GPU in local device 2x 15 GB T4 GPU
Hugging Face	Model & Dataset Collection	Kaggle	Easy to select based on our requirements
Chat GPT	Compare summaries, bug fixing support	Stack Overflow	To debug & compare outputs
Transformers (Happy Transformer)	Library for Python		Open-source and offers a complete set of tools and pre-trained models for dealing with a variety of NLP applications
<i>google/pegasus-cnn_dailymail</i>	Model used to train and generate summary		
<i>EleutherAI/gpt-neo/2.7B</i>	Model used to Fine-tune and generate summary		Previously selected PEGASUS model didn't work
<i>Microsoft/phi-2</i>	Model Used to generate summary using zero shot and few shot		Previously selected model couldn't generate a proper summary of meeting script
<i>Prompt Engineering</i>	Using RAG Technique		Improve the generated Summary

### 3.3

## Software Implementation

### *Google/pegasus-cnn\_dailymail*

We train the pegasus model. The training process of Pegasus involves customizing a pre-trained model to perform a specific task, such as summarizing news articles in the case of CNN-DailyMail. This customization includes fine-tuning the model with a task-specific head, defining relevant loss functions, and training it on a specific dataset. Hyperparameter tuning is performed to optimize the model's performance. Regular validation and testing are crucial to monitor and ensure the model's quality. If necessary, post-processing steps can be applied to enhance the generated results. Finally, once the model performs satisfactorily, it can be deployed for the intended task.

```
print("\nDialogue: ")

print(dataset_samsum["test"][0]["dialogue"])
dialogue = dataset_samsum["test"][0]["dialogue"]

print("\nSummary from the dataset: ")
print(dataset_samsum["test"][0]["summary"])

pipe = pipeline('summarization', model= model_hf)

pipe_out = pipe(dialogue)
print("\nDirect output from the model: \n")
print(pipe_out)

print("\nActual Summary Below: \n")
print(pipe_out[0]['summary_text'].replace(" .<n>"," .\n"))
```

Figure 8: Code snippets using *pipe*

Implementing the hugging face model using *pipe*

```
1 from transformers import TrainingArguments, Trainer
2
3 trainer_args = TrainingArguments(output_dir='/content/drive/MyDrive/Samsun_Pegasus_New_Train/pegasus-samsum',
4                                 num_train_epochs=2, warmup_steps=500, per_device_train_batch_size=1, per_device_eval_batch_size=1,
5                                 weight_decay=0.01, logging_steps=10, evaluation_strategy='steps', eval_steps=500,
6                                 save_steps=1e6, gradient_accumulation_steps=16, learning_rate=1e-4, report_to="tensorboard")
```

Figure 9: Code snippets of training arguments

```
1 trainer = Trainer(model=pegasus_model, args=trainer_args,
2                     tokenizer=tokenizer, data_collator=seq2seq_data_collator,
3                     compute_metrics = compute_metrics,
4                     train_dataset = dataset_samsum_pt["train"],
5                     eval_dataset = dataset_samsum_pt["validation"])
```

Figure 10: Code snippets of *trainer*

Using this portion of code, we ***fine-tuned*** the previously selected ***Pegasus-CNN-Dailymail*** model. According to these criteria, the model is fine-tuned and trained up to **15 epochs** and Tested with **12 fields** ( paper of Pegasus model) related data.

## **EleutherAI/get-neo-2.7B + EleutherAI/get-neo-125M**

Here, we load the model by a transformer model called “***Happy Transformer***”. We load the model as a generator. It means load with HappyGenerator() function [Code in below]. Firstly, we load this gpt-neo model with 2.7 Billion parameters. It was loaded successfully but unfortunately due to it’s huge size it couldn’t be able to train (cuda memory broke down with 2x T4-GPU and 29 GB RAM). However, for this problem we again try with this model but this time with 125 Million Parameters for training with cnn dataset and HappyGenerator. Then we tried to train with 1 epoch. It was successfully loaded again and trained [Code snippets and screenshot below].

We tested the train set first with a customized dataset.

***How we customize dataset for prompt:*** There are 3 columns in the datasets. Given below:

[train.csv \(1.26 GB\)](#)

Detail   Compact   Column

About this file

287,113 unique articles were provided for train.

<b>id</b>	<b>article</b>	<b>highlights</b>
heximal formatted SHA1 hash of the url where the story was retrieved from	a string containing the body of the news article	a string containing the summary of the article
<b>287113</b> unique values	<b>284005</b> unique values	<b>282197</b> unique values
0001d1afc246a7964130f43ae940af6bc6c57f01	By . Associated Press . PUBLISHED: . 14:11 EST, 25 October 2013 .   .	Bishop John Folla, of North Dakota, is taking time off after being

**Figure 11: Dataset Summary**

We merged it in just 1 column. The input text: in the first then summary:  
like ***Text <> Summary: <>***

So that, the result will be a summary from the given text. Therefore, the training process with 1 epoch was done.

[train25000.csv \(109.51 MB\)](#)

Detail   Compact   Column

**text**

**24983**  
unique values

(CNN) -- Ralph Mata was an internal affairs lieutenant for the Miami-Dade Police Department, working...

A drunk driver who killed a young woman in a head-on crash while checking his mobile phone has been ...

The actual dataset size is huge (almost 1.6 GB) which break the training System. For this reason we split the dataset with 3 segments. Firstly, split with 2500 cells, then 25000 cells, then 50000 cells, the 100000 cells. But again it breaks for 50000 and 100000 cell train sets. Therefore, we worked with 25000 cell dataset that we split before. Then we trained. The dataset now like this with one column. ***Text <> Summary: <>***

## Load the Happy Transformer with EleutherAI/gpt-neo-125M:

```
[3] from happytransformer import HappyGeneration  
  
[4] happy_gen = HappyGeneration("GPT-NEO", "EleutherAI/gpt-neo-125M")  
  
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:  
The secret `HF_TOKEN` does not exist in your Colab secrets.  
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),  
You will be able to reuse this secret in all of your notebooks.  
Please note that authentication is recommended but still optional to access public models or datasets.  
    warnings.warn(  
config.json: 100% [00:00<00:00, 59.6kB/s]  
model.safetensors: 100% [00:03<00:00, 162MB/s]  
generation_config.json: 100% [00:00<00:00, 4.22kB/s]  
tokenizer_config.json: 100% [00:00<00:00, 33.1kB/s]  
vocab.json: 100% [00:00<00:00, 3.58MB/s]  
merges.txt: 100% [00:00<00:00, 2.25MB/s]  
tokenizer.json: 100% [00:00<00:00, 11.8MB/s]  
special_tokens_map.json: 100% [00:00<00:00, 16.4kB/s]
```

Figure 12: Happy Transformer Loading

## Train for 25000 cell dataset:

```
Run All | Code ▾  
▶ happy_gen.train('/kaggle/input/train25000/train25000.csv')  
  
Generating train split: [00:03<00:00, 8022.04 examples/s]  
Map: 100% [00:49<00:00, 458.67 examples/s]  
Map: 100% [00:05<00:00, 460.14 examples/s]  
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning:  
alar; will instead unsqueeze and return a vector.  
    warnings.warn('Was asked to gather along dimension 0, but all ']  
[11250/11250 4:27:28, Epoch 1/1]  


| Step  | Training Loss | Validation Loss |
|-------|---------------|-----------------|
| 1     | 5.754000      | 6.869854        |
| 1125  | 7.178200      | 8.763229        |
| 2250  | 4.963600      | 4.712391        |
| 3375  | 4.587700      | 5.139930        |
| 4500  | 4.743800      | 4.967206        |
| 5625  | 5.726100      | 4.385664        |
| 6750  | 6.027700      | 20.808872       |
| 7875  | 6.656600      | 4.411877        |
| 9000  | 5.432300      | 6.063212        |
| 10125 | 6.019000      | 5.828940        |
| 11250 | 5.525100      | 5.393856        |

  
+ Code + Markdown  
1]: args2 = GENSettings(min_length=70, max_length=100)  
result = happy_gen.generate_text("Liverpool target Neto is also wanted
```

Figure 13: Happy Transformer Loading

## Phi-2

Firstly, we import a text generation model from the Hugging Face Transformers library and initialize it with a specified model called "microsoft/phi-2". The pipeline function sets up a text generation pipeline with the specified model. Then generates a text summary based on the provided prompt using a text generation model. The summary is presented using Markdown format. Then initializes a tokenizer for the "phi-2" model with a causal language model ("AutoModelForCausalLM") for the "microsoft/phi-2" model, with automatic specification of torch data type and device mapping. Note that the model is pretrain. So, we didn't train for it.

Finally, we generate a short summary of the given prompt using the initialized language model. It utilizes the tokenizer to encode the prompt and the model to generate the summary based on the encoded prompt. The summary is then decoded and printed.

Most importantly, we tried the **zero shot technique** here to generate a summary. We'll try **few shot techniques** for a more accurate summary from a text. After that from a long meeting we'll generate summary and highlighting points.

### **Load the model via pipeline** (Zero shot Technique):

```
from transformers import pipeline

model_name = "microsoft/phi-2"

pipe = pipeline(
    "text-generation", #s
    model=model_name,
    device_map="auto",
    trust_remote_code=True,
)

config.json: 100% [863/863 [00:00<00:00, 68.8kB/s]

configuration_phi.py: 100% [9.26k/9.26k [00:00<00:00, 840kB/s]

A new version of the following files was downloaded from https://huggingface.co/microsoft/phi-2:
- configuration_phi.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloading new versions of the code file, you can pin a revision.

modeling_phi.py: 100% [62.7k/62.7k [00:00<00:00, 772kB/s]

A new version of the following files was downloaded from https://huggingface.co/microsoft/phi-2:
- modeling_phi.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloading new versions of the code file, you can pin a revision.

modelsafetensors.index.json: 100% [35.7k/35.7k [00:00<00:00, 3.24MB/s]

Downloading shards: 100% [2/2 [00:59<00:00, 24.87s/it]

model-00001-of-00002.safetensors: 100% [5.00G/5.00G [00:57<00:00, 386MB/s]

model-00002-of-00002.safetensors: 100% [564M/564M [00:01<00:00, 385MB/s]

Loading checkpoint shards: 100% [2/2 [00:09<00:00, 4.22s/it]

generation_config.json: 100% [124/124 [00:00<00:00, 11.4kB/s]

tokenizer_config.json: 100% [7.34k/7.34k [00:00<00:00, 632kB/s]

vocab.json: 100% [798k/798k [00:00<00:00, 29.5MB/s]

merges.txt: 100% [456k/456k [00:00<00:00, 27.2MB/s]

tokenizer.json: 100% [2.11M/2.11M [00:00<00:00, 6.38MB/s]

added_tokens.json: 100% [1.08k/1.08k [00:00<00:00, 90.3kB/s]

special_tokens_map.json: 100% [99.0/99.0 [00:00<00:00, 7.65kB/s]

Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.

+ Code + Markdown
```

Figure 14: Phi-2 Loading

### Tokenizer and causal language model:

```
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

tokenizer = AutoTokenizer.from_pretrained(
    "microsoft/phi-2",
    trust_remote_code = True
)

model = AutoModelForCausalLM.from_pretrained(
    "microsoft/phi-2",
    torch_dtype = "auto",
    device_map = "auto",
    trust_remote_code = True
)
```

Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.

Loading checkpoint shards: 100% 2/2 [00:02<00:00, 1.17s/it]

Figure 15: Phi-2 Loading Tokenizer

### Output Code:

```
prompt = "Give me a short summary of the following text. Asana and Avir are twins who j  
with torch.no_grad():  
    token_ids = tokenizer.encode(prompt, add_special_tokens=False ,return_tensors="pt")  
    output_ids = model.generate(  
        token_ids.to(model.device),  
        max_new_tokens=512,  
        do_sample=True,  
        temperature = 0.3  
    )  
  
    output = tokenizer.decode(output_ids[0][token_ids.size(1) :])  
    print(output)
```

Figure 16: Phi-2 Output Code

## RAG Technique

We tried to load the requires libraries needed to use RAG technique of prompt engineering. We mainly focused on LangChain and ChromaDB library for this. So, we import those libraries.



A screenshot of a Jupyter Notebook interface. The sidebar on the left shows various icons for file operations. The main area displays a code cell with the following imports:

```
from torch import cuda, bfloat16
import torch
import transformers
from transformers import AutoTokenizer
from time import time
#import chromadb
#from chromadb.config import Settings
from langchain.llms import HuggingFacePipeline
from langchain.document_loaders import TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.chains import RetrievalQA
from langchain.vectorstores import Chroma
```

We tried to load our model `microsoft/phi-2` using those required libraries of RAG. We used CNN-DailyMail News dataset for generating vectors for the context.



A screenshot of a Jupyter Notebook interface. The sidebar on the left shows various icons for file operations. The main area displays two code cells. The first cell contains code to set up the model configuration:

```
model_id = 'microsoft/phi-2'

device = f'cuda:{cuda.current_device()}' if cuda.is_available() else 'cpu'

# set quantization configuration to load large model with less GPU memory
# this requires the 'bitsandbytes' library
bnb_config = transformers.BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type='nf4',
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=bfloat16
)
```

The second cell contains code to initialize the tokenizer:

```
tokenizer = AutoTokenizer.from_pretrained("microsoft/phi-2", trust_remote_code=True)
```

Lastly, we imported the tuned tokenizer from the `microsoft/phi-2` and tried to work with that.



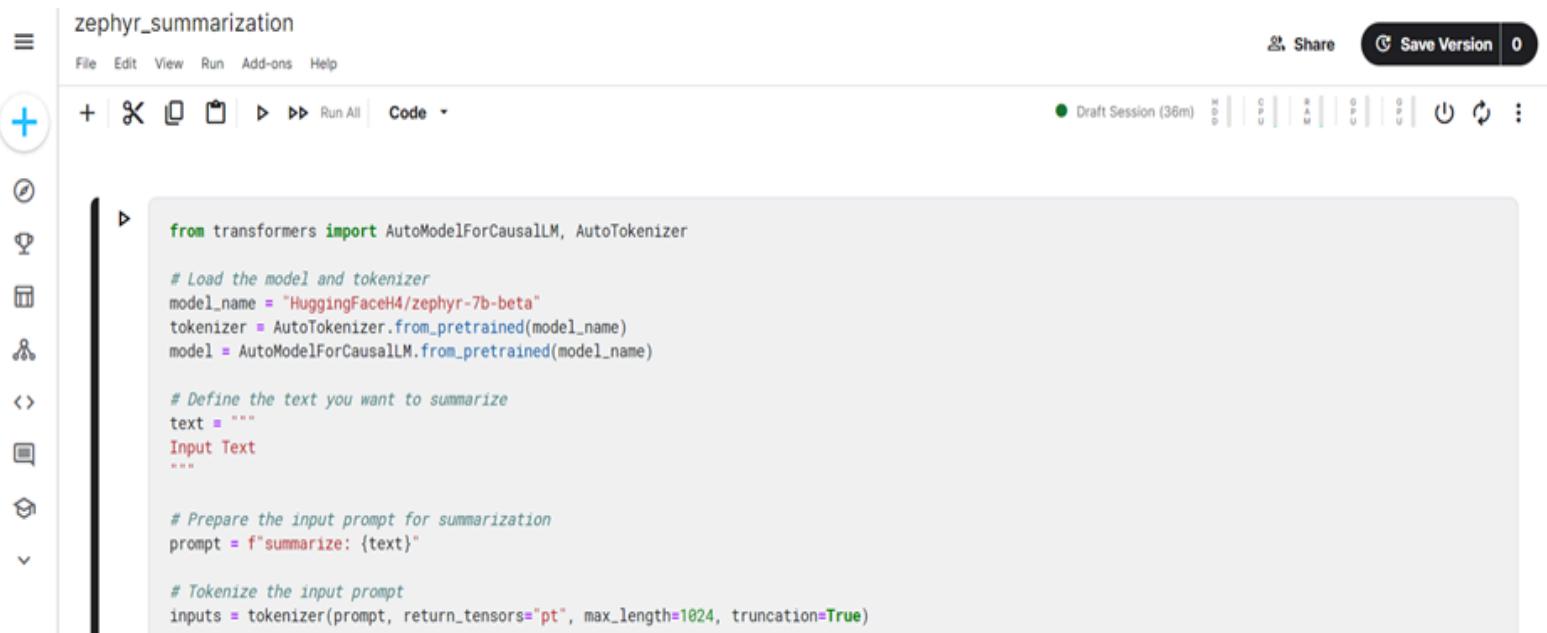
A screenshot of a Jupyter Notebook interface. The sidebar on the left shows various icons for file operations. The main area displays a code cell with the following code:

```
[39]: time_1 = time()
model_config = transformers.AutoConfig.from_pretrained(
    model_id
)
model = transformers.AutoModelForCausalLM.from_pretrained(
    model_id,
    trust_remote_code=True,
    config=model_config,
    quantization_config=bnb_config,
    device_map='auto',
)
tokenizer = AutoTokenizer.from_pretrained(model_id)
time_2 = time()
print(f"Prepare model, tokenizer: {round(time_2-time_1, 3)} sec.")
```

**Figure 17: RAG implement (all 3)**

## Zephyr

We used another model named HuggingFaceH4/zephyr-7b-beta and tried to use this in prompt engineering technique. So, first we imported the required library for that like tokenizer, AutoModel library etc. By using these libraries we can load the attributes or elements of the fine-tuned Zephyr model in our notebook to generate summaries from the long scripts.



```
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load the model and tokenizer
model_name = "HuggingFaceH4/zephyr-7b-beta"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Define the text you want to summarize
text = """
Input Text
"""

# Prepare the input prompt for summarization
prompt = f"summarize: {text}"

# Tokenize the input prompt
inputs = tokenizer(prompt, return_tensors="pt", max_length=1024, truncation=True)
```

Then, we tried to load the tokenizer and other elements of the Zephyr model in our notebook.



```
# Generate the summary
summary_ids = model.generate(
    inputs["input_ids"],
    max_length=150,
    min_length=30,
    length_penalty=2.0,
    num_beams=4,
    early_stopping=True
)

# Decode the summary
summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

print("Original Text:", text)
print("\nSummary:", summary)
```

**Figure 18: Zephyr implement and output**

## Chapter 4 Investigation/Experiment, Result, Analysis, and Discussion

### Google/pegasus-cnn\_dailymail

**Investigation/Experiment:** Our supervisor asked us to check the maximum length of input that is allowed by the Pegasus CNN dailymail model. We tried with a couple of long texts as our meeting script will be so long but found out the model only takes a script with 1200 words or less as an input. This was a problem for the Pegasus CNN dailymail model which was solved when we used Microsoft phi-2 model. Because it can generate output for a very long script. So, the Phi-2 model was a lifesaver for us this time.

**Result and Analysis:** According to our *pegasus cnn-dailymail* model, we fine-tune the model and get a result of dialogue summarization. From the samsum dataset, we take the input of the main text and summarize it in our model. The model gives results like the following:

```
Using Pegasus to make the summary using DL, using Pipeline

print("\nDialogue: ")
print(dataset_samsum["test"][0]["dialogue"])
dialogue = dataset_samsum["test"][0]["dialogue"]

print("\nSummary from the dataset: ")
print(dataset_samsum["test"][0]["summary"])

pipe = pipeline('summarization', model= model_hf)

pipe_out = pipe(dialogue)
print("\nDirect output from the model: \n")
print(pipe_out)

print("\nActual Summary Below: \n")
print(pipe_out[0]['summary_text'].replace(" .<n>"," .\n"))

Dialogue:
Hannah: Hey, do you have Betty's number?
Amanda: Lemme check
Hannah: <file_gif>
Amanda: Sorry, can't find it.
Amanda: Ask Larry
Amanda: He called her last time we were at the park together
Hannah: I don't know him well
Hannah: <file_gif>
Amanda: Don't be shy, he's very nice
Hannah: If you say so..
Hannah: I'd rather you texted him
Amanda: Just text him 😊
Hannah: Urgh.. Alright
Hannah: Bye
Amanda: Bye bye

Summary from the dataset:
Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.
Some weights of PegasusForConditionalGeneration were not initialized from the model checkpoint at google/pegasus-cnn_dailymail.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Your max_length is set to 128, but your input_length is only 122. Since this is a summarization task, where outputs shorter than 128 tokens are common, this is likely fine.

Direct output from the model:
[{'summary_text': 'Amanda: Ask Larry Amanda: He called her last time we were at the park together .<n>Hannah: I\'d rather you texted him .<n>Amanda: Just text him .'}

Actual Summary Below:

Amanda: Ask Larry Amanda: He called her last time we were at the park together .
Hannah: I'd rather you texted him .
Amanda: Just text him .
```

Figure 19: Code snippets of generating summary from the samsum dataset before train

Here is the generated dialogue summary from the samsum dataset text input using DL and pipeline. The summary is not so good after generating it.

```

1 sample_text = "Liam: <file_other> Liam: have you read this one? Julia: wow, I didn't even know that it existed Julia: thanks! The last o
2
3 print("\nNew Trained Model Summary [24]:\n")
4 print(pipe(sample_text, **gen_kwargs)[0]["summary_text"])

Your max_length is set to 256, but your input_length is only 87. Since this is a summarization task, where outputs shorter than the input are
New Trained Model Summary [24]:
Julia has read a book from 2014 and recommends it to Liam. The city library is organizing a meeting with Tess. Liam suggests checking it out
< />

1 sample_text ="Mia: babe, do you mind if i go out after work tonight? Elliot: yeah, sure. go for it ;) Mia: thanks, babe. it's gonna be g
2
3 print("\nNew Trained Model Summary [25]:\n")
4 print(pipe(sample_text, **gen_kwargs)[0]["summary_text"])

Your max_length is set to 256, but your input_length is only 90. Since this is a summarization task, where outputs shorter than the input are
New Trained Model Summary [25]:
Mia is going out with Elliot tonight. They're going to grab some drinks and have some nice time outside of work. Elliot will pick Mia up.

```

**Figure 20: Code snippets of generating summary from the samsum dataset after train**

Here is a short sample of the trained model summary.

We compare it with the ChatGPT summary and hugging face API summary after that. However, our model-generated summary had better results in some cases, whereas other summaries of chatGPT or hugging face API got good results sometimes, according to our analysis. We compare 55 test cases among all those summaries with the main test.

The comparison results are given below:

### **Comparison Between Hugging Face Summarization and Dataset Summarization**

(CSE499A\_Section 10.29\_Autonote\_Meeting Summarization\_Md Saiyem Raiyan, Samia Sultana, Sheikh Mohammed Wali Ullah, Zobaer Ahammud Zamil)

No	Main Text	Hugging Face API Says	Dataset Summary	GPT 3.5 Summary	New Trained Model Summary
1	"Hannah: Hey, do you have Betty's number? Amanda: Lemme check Hannah: <file_gif> Amanda: Sorry, can't find it. Amanda: Ask Larry Amanda: He called her last time we were at the park together Hannah: I don't know him well Hannah: <file_gif> Amanda: Don't be shy, he's very nice Hannah: If you say so.. Hannah: I'd rather you texted him Amanda: Just text him 😊 Hannah: Ugh.. Alright Hannah. Bye Amanda: Bye bye"	Hannah: I'd rather you texted him. Hannah: I'd rather you texted him.Amanda: Just text him .	"Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry."	Hannah asks Amanda for Betty's number, but Amanda can't find it. She suggests asking Larry, who called Betty last time they were at the park. Hannah hesitates, but Amanda assures her Larry is nice. Hannah eventually agrees but prefers Amanda to text him. They say their goodbyes.	Amanda can't find Betty's number. Larry called her the last time they were at the park together. Hannah wants Amanda to text Larry instead.
2	"Eric: MACHINE! Rob: That's so gr8! Eric: I know! And shows how Americans see Russian :) Rob: And it's really funny! Eric: I know! I especially like the train part! Rob: Hahaha! No one talks to the machine like that! Eric: Is this his only stand-up? Rob: Idk. I'll check. Eric: Sure. Rob: Turns out no! There are some of his stand-ups on youtube. Eric: Gr8! I'll watch them now! Rob: Me too! Eric: MACHINE! Rob: MACHINE! Eric: TTYL? Rob: Sure :)"	Rob tries to keep up with Eric's Russian jokes . Eric tries to keep up with Rob's Russian jokes. Rob tries to keep up with Eric's Russian jokes .	"Eric and Rob are going to watch a stand-up on youtube."	Eric and Rob share excitement about a stand-up comedy performance, appreciating its humor and a particular train-related part. They discuss whether it's the comedian's only stand-up, with Rob checking and finding more on YouTube. They both plan to watch more of his performances and say goodbye.	Eric and Rob are going to watch some of the Russian comedian's stand-ups. They like the one about the machine and think it's funny.
11	"Adam: Have you talked to May? Karen: Yes, yesterday, why? Adam: I just talked to her and I must admit I worry about her Karen: Me too, I suggested she should see a specialist, but she wasn't very happy about it Adam: No wonder... Karen: I know, but I think this is serious. She's saying she's depressed, like everyone around, but in her case it may be true Adam: She was telling me she doesn't feel like doing anything, she's bored all the time, she never feels happy. It sounds like a real, typical depression Adam: She also told me that she has trouble sleeping. I asked her to go out for a beer or anything basically, but she doesn't want to leave the flat Karen: Oh my, it sounds really serious. I don't know what to tell you Adam: I was wondering how I can help her Karen: Honestly I don't know if we can help her, Adam. I suggested a specialist because these are very sensitive issues and I'm afraid we may unintentionally make it worse Adam: Yes, but she doesn't want to see a specialist. Basically, she doesn't want to see anyone Karen: Hm... I don't know... How about I call someone for advice? So we could know what to do Adam: Sounds rational, do you know anyone you could call? Don't mention her name Karen: Of course I won't! I have a friend who's a psychologist, we can trust her. I'll let you know Adam: Thank you Karen!"	May says she doesn't feel like doing anything, she's bored all the time and never feels happy .Adam suggests she see a specialist, but she doesn't want to see anyone .	"Adam and Karen are worried that May suffers from depression. Karen will call her friend who is a psychologist and ask for advice. "	Adam and Karen discuss their concerns about their friend May, who is showing signs of depression. They both agree that May's condition seems serious. Adam mentions that May has trouble sleeping and doesn't want to leave her flat. Karen suggests seeking advice from a psychologist friend without mentioning May's name, and Adam agrees to this plan, expressing gratitude for Karen's help and concern.	May is depressed. She doesn't want to see a specialist. Karen has a friend who's a psychologist. She will call her for advice.
12	"Augustine: Guys, remember it's Wharton's bday next week? Darlene: yay, a party! Heather: yay! crap we need to buy him a present Walker: he mentioned paper shredder once Augustine: wtf?? Walker: He really did, for no reason at all. Heather: Whatever that makes him happy Darlene: cool with me. we can shred some papers at the party Augustine: so much fun Heather: srslly guys, you mean we should really get office equipment??? Darlene: Walk, ask him if he really wants it and if he yes then we get it Walker: i heard him say that. wasn't drunk. me neither. Darlene: But better ask him twice. Walker: Will Augustine: 2moro ok? Darlene: And sure, ask about the party!"	Walker: he mentioned paper shredder once Augustine: wtf??Darlene: but better ask him twice Walker: will do Augustine: 2moro ok?Heather: whatever that makes him happy Darlene: cool with me. we can shred some papers at the party Augustine: so much fun .	"Next week is Wharton's birthday. Augustine, Darlene, Heather and Walker want to buy him a paper shredder. Walker will make sure if Wharton really wants it. "	Augustine reminds the group about Wharton's upcoming birthday. They discuss getting him a paper shredder, even though it seems unusual. Heather and Darlene are on board, but Heather questions the idea of buying office equipment. Darlene suggests asking Wharton again to confirm his preference, and Walker agrees to do so tomorrow. They also plan to ask about the party.	It's Wharton's birthday next week. Darlene, Augustine, Heather and Walker are throwing a party. Darlene wants to get office equipment for Wharton.

**Figure 21: Generated summary comparison with ChatGPT summary**

We finally analyzed about 12 fields that were mentioned in the paper of the Pegasus model. After testing the 12 field data of the Pegasus model, it cannot take input over a 1024-length string. In this case, we analyze a long transcript result.

In summary:

- a. Meeting transcript downloaded using various methods from
  - i. Particular website
  - ii. youtube

b. Flaw: A string greater than 1024 in length cannot be inserted into the model

Therefore, we solved this by applying a methodology. Firstly, the main text breaks down into segments. For example, A text of 3000 words is split into three parts. Each split takes 1000 lengths of text. Then, summarize each split and finally merge it to get a result. So, a summary of the concatenated summary of previous segments. Here is an example.

**Summary:** Pitambaran's wife and daughter died from a deadly disease. Pitambaran lost a lot of weight and decided to work as a laborer and make some money for his family. Pitambaran worked in the fields and begged for food in the streets of the village. Pitambaran butchered a goat and put it on the top of the farm.

[Split 1]

**Summary:** Pitambaran stole a goat from his neighbor's house. The leader of the village assigned a few Gods to search around the area nearby the farm of Ravi so they could find who had stolen the goat. The guards had found a bag of meat on top of Winans Farm. Pitambaran's father is a witness to the entire incident.

[Split 2]

**Summary:** Pitambaran and his father are allowed to leave right now both pitambaran and his father fold their hands towards the judge and they thank him for what he had done for them.

[Split 3]

### Final Summary:

#### TOTAL SUMMARY (Model)

Pitambaran's wife and daughter died from a deadly disease. Pitambaran lost a lot of weight and decided to work as a labor and make some money for his family. Pitambaran butchered a goat and put it on the top of the farm.

## Results:

**Rouge-Score:** It measures the quality of the text summarization component. These scores evaluate how well the system-generated meeting summaries align with human-generated reference summaries. It's a key metric for assessing the effectiveness of our system in capturing essential meeting content. It offers the advantages of ease of calculation and interpretation, making it a valuable tool for assessing summaries.

## Before train

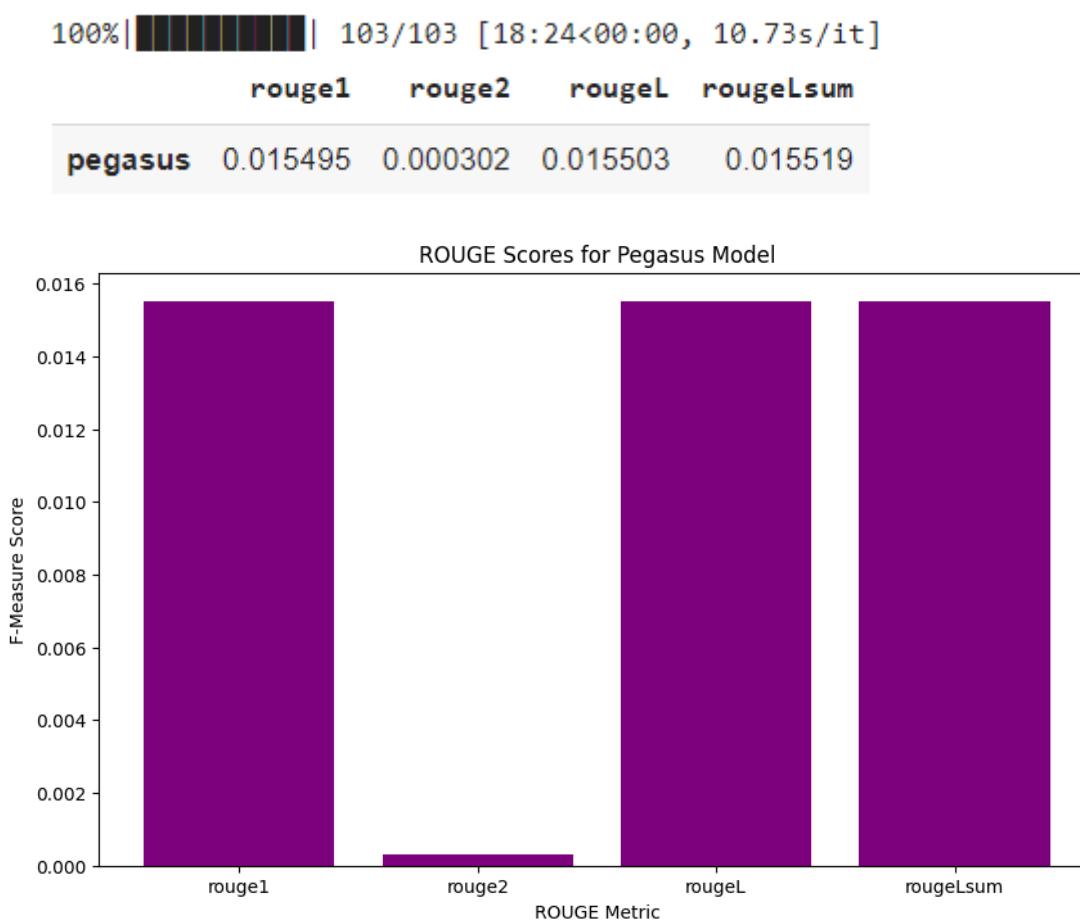


Figure 22: ROUGE score before training and graph

## After 15 Epoch

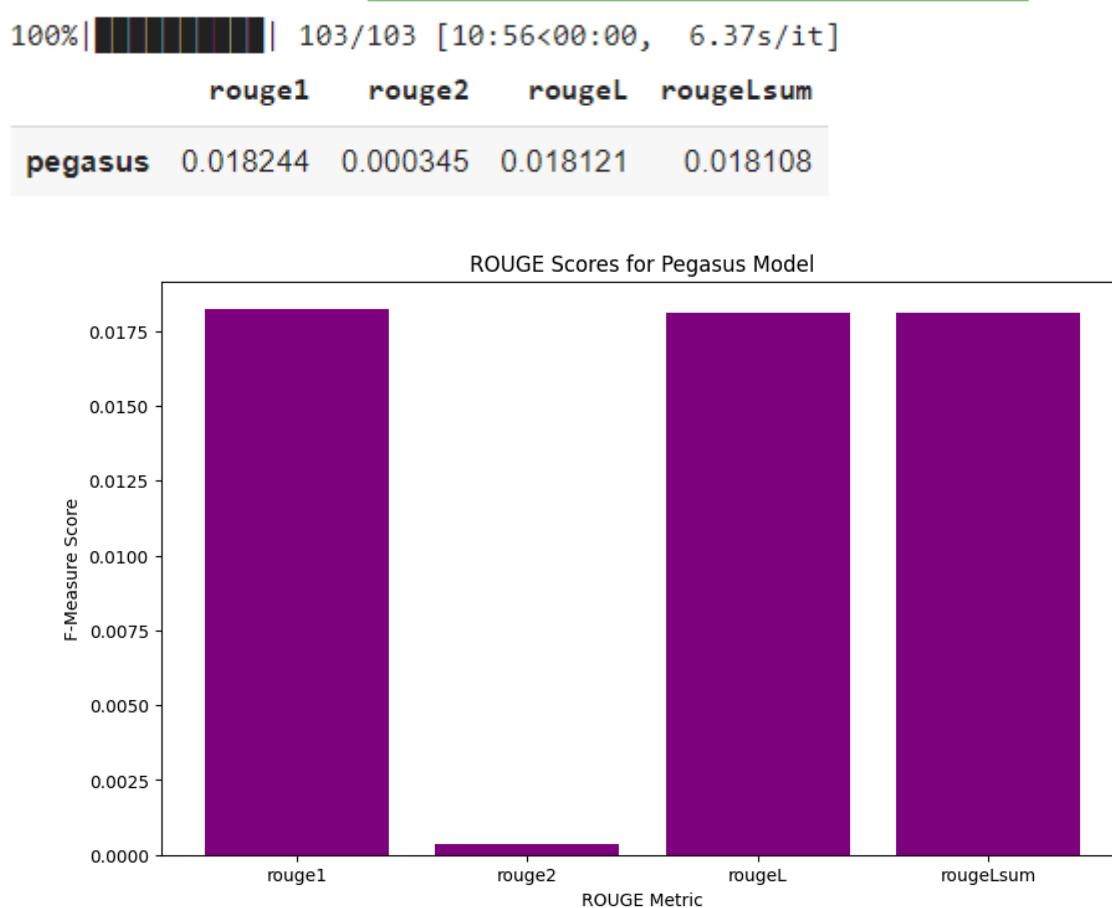


Figure 23: ROUGE score after training and graph

**Discussion:** According to our pegasus model, the summary generated by the model is not well. Sometimes, it gives a short summary of the pegasus model, which is good. However, in most cases, it cannot give the meaningful summary that ChatGPT gives. 12 field testing from the pegasus model is good in some cases, but with a huge and long transcript, it cannot give a meaningful summary. So, the model cannot work efficiently. Thus, the summary cannot work well in this format according to the pegasus model. So, by analyzing the new model, we found the **EleutherAI/get-neo-2.7 B** model, a transformer model designed using EleutherAI's replication of the GPT-3 architecture. GPT-Neo refers to the class of models, while 2.7B represents the number of parameters of this particular pre-trained model. The EleutherAI model works more efficiently than pegasus cnn-dailymail.

[EleutherAI/get-neo-2.7B](#) + [EleutherAI/get-neo-125M](#)

**Investigation/Experiment:** The investigation and experiment about this model through reading a lot of papers and youtube videos. Most importantly, our supervisor advised us to load the model using “Happy Transformer” which is a technique to load fine-tuned transformer models. This has shown promising results, particularly when fine-tuned and properly configured. However, challenges such as bias, handling long texts, and domain-specific adaptation remain areas for ongoing research and improvement. The models have large parameters. We tried several approaches by using 125M, 1.3B and 2.7 Billion Parameters. In 1.3B and 2.7B Parameters there are huge loading issues due to local memory in fact online notebook memory. 1.3B Parameters were loaded but not trained. Therefore, at last the 125M Parameters model had been loaded and trained with 1 epoch with a lot of time but couldn’t give the contextual summary. The Happy Transformer model was not loaded with HappyTextToText but we are able to load with HappyTextGenerator. Finally, The model was trained but redundant and not given contextual and meaningful summary.

**Result and Analysis:** After several training, the model gives a redundant summary. Firstly, it gave a summary but not meaningful. Lastly, it gave an incoherent summary. Sample given below:

Beirut (CNN) -- Syria carried out an airstrike on a refugee camp in northern Lebanon Saturday, killing nine Syrians and wounding nine more, a Lebanese state-run news agency reported. The strike centered on a Syrian refugee camp located near the Syrian border between the towns of Baalbeck and Arsal in the Bekaa Valley, the National News Agency said. The Red Cross took the casualties to Universal Hospital in Baalbek. Saturday's strike was not the first by the Syrian government, which has accused rebels of smuggling arms and supplies across the border. On March 18, two Syrian jets fired three rockets that hit empty buildings near Arsal. At the time, a U.S. State Department spokeswoman called the use of fighter jets to fire rockets into Lebanon a "significant escalation." U.N. commissioner wants to probe into whether Syrian rebels executed soldiers . Also in March, the U.N. Security Council voiced "grave concern over repeated incidents of cross-border fire which caused death and injury among the Lebanese population, incursions, abductions and arms trafficking across the Lebanese-Syrian border, as well as other border violations." The declaration followed a briefing by officials on how the conflict in Syria has spilled into Lebanon. More than 600,000 Syrians have fled to neighboring Lebanon, a country of about 4 million people, according to a U.N. estimate. But the Lebanese government puts the total at more than 1 million. Whatever the true figure, there is no dispute that the influx has destabilized the area and heightened tensions. The attack comes as the Syrian conflict is mired in a third year of unrest, which started in March 2011 when President Bashar al-Assad cracked down on peaceful protesters. Since then, it has evolved into a civil war that has killed more than 100,000 and transformed more than 1 million others into refugees, according to the Red Cross. Read more: U.N. inspectors heading to Syria to probe chemical weapons reports . CNN's Nick Paton Walsh reported this story from Beirut, and Tom Watkins wrote it in Atlanta. CNN's Hamdi Alkhshali and Yousuf Basil contributed to this report . Summary: Airstrike kills nine Syrians in refugee camp, state media reports .

Syria has fired into Lebanon before .

The government has accused rebels of smuggling arms across the border with Lebanon .

**Training Results:** Invalid text [Output given below]

```
args2 = GENSettings(min_length=70, max_length=100)
result = happy_gen.generate_text("A thief nicknamed the 'Black Widow of Facebook' who drugged and robbed men after she lured them into bed has bee
```

Setting `pad\_token\_id` to `eos\_token\_id`:50256 for open-end generation.

*Figure 24: ElutherAI/gpt-neo training results*

**Discussion:** According to *EleutherAI/gpt-neo model and happy transformer* the summary can not be able to work. This gives redundant output, the same token repeated again and again [screenshot above]. Anyhow it can not generate any good summary. For this major problem we again shift with a new model called “Phi-2” from microsoft.

## ***Phi-2 with Zero shot and Few shot***

**Investigation/Experiment:** OpenAI couldn't generate the summary for a specific category of the text but the phi-2 model really saved us this time. Or else, we need to find a new model from hugging face to generate the summary and wait for the comparison to happen. General *Microsoft-Phi-2* model generate a good summary but we tried to improve those by Prompt Engineering Techniques which are Zero shot and Few Techniques. Those techniques are very beneficial for generating a good summary. Our supervisor asked us to compare how the change of maximum token number and temperature affects the generated summary. We tried with different no. of tokens, but at last, we came to a conclusion to use 30% token of the whole text. Moreover, for the temperature, we considered 0.3 as an optimum value because if we change it (raise or lower), we can see that the summary of all types of text is not okay with that. So, in this case, temperature value 0.3 and 30% max token value were the lifesaver for us. Otherwise, we need to check for the whole values range which is a very complicated task. Finally, We experiment summaries with Hemingway and Grammarly Premium websites. The summaries gave best scores by comparing with OpenAI (ChatGPT and Gemini)

**Result and Analysis:** As we mentioned, our results were good after using Prompt Engineering Techniques.

```
[15]:  
prompt = "Give me a short summary of the following text. One hot day, a thirsty crow flew all over the fields looking for water. He was very thirsty and found a jug of water. But the jug had a narrow neck, so he couldn't drink from it. He thought for a moment and then decided to use some pebbles to raise the water level and quench his thirst."  
with torch.no_grad():  
    token_ids = tokenizer.encode(prompt, add_special_tokens=False, return_tensors="pt")  
    output_ids = model.generate(  
        token_ids.to(model.device),  
        max_new_tokens=512,  
        do_sample=True,  
        temperature = 0.3  
    )  
    output = tokenizer.decode(output_ids[0][token_ids.size(1) :])  
    print(output)  
  
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.  
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.  
OUTPUT: A thirsty crow found a water jug but couldn't drink from it due to its narrow neck. He used pebbles to raise the water level and quench his thirst.  
<|endoftext|>  
  
[16]:  
prompt = "Give me a short summary of the following text. Hare was much amused at the idea of running a race with the Tortoise. He challenged the tortoise to a race. The hare is overconfident and lazy, while the tortoise is slow but steady. The hare wastes time by sleeping, while the tortoise keeps moving. The tortoise wins the race and teaches the hare a lesson."  
with torch.no_grad():  
    token_ids = tokenizer.encode(prompt, add_special_tokens=False, return_tensors="pt")  
    output_ids = model.generate(  
        token_ids.to(model.device),  
        max_new_tokens=512,  
        do_sample=True,  
        temperature = 0.3  
    )  
    output = tokenizer.decode(output_ids[0][token_ids.size(1) :])  
    print(output)  
  
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.  
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.  
ANSWER: The text is a fable about a race between a hare and a tortoise. The hare is overconfident and lazy, while the tortoise is slow but steady. The hare wastes time by sleeping, while the tortoise keeps moving. The tortoise wins the race and teaches the hare a lesson.  
<|endoftext|>
```

**Figure 25: phi-2 results with prompt engineering techniques**

## Phi-2 with RAG Technique

**Investigation/Experiment:** As we are facing issues with fine-tuning due to the resource limitations, our supervisor asked us to use prompt engineering technique with a small LLM model and suggested us microsoft/phi-2 model. Another, As prompt engineering like zero shot & few shot were working good with Microsoft/phi-2, we tried to use the retrieval augmented generation technique (RAG) to increase its performance so that it can show better summaries with more concise points. We started this with the langchain and chromaDB library as we want the generate a vector dataset from our used dataset “CNN-DailyMail-News-Dataset”. After that, we tried to make a vector dataset where we could search for the context of our input every time we input anything. So, in this case we have extra information at input that is the context vector of the input.

**Result and Analysis:** However, we couldn't manage to generate that vector database from our dataset as for the resource convenience we used Kaggle notebook for this attempt. Also, Kaggle doesn't have the Microsoft/phi-2 model in the collection. So, we couldn't input the model from Kaggle. Again, we tried the same thing in colaboratory but couldn't find any solution for that. So, this was our RAG approach but we failed to generate a summary from this.



The screenshot shows a Jupyter Notebook interface with a single code cell (In[39]) containing Python code. The code is intended to initialize a model and measure the time taken. However, it fails with a KeyError because the 'model\_id' key is missing from the 'AutoConfig' object. The error message is displayed below the code cell.

```
time_1 = time()
model_config = transformers.AutoConfig.from_pretrained(
    model_id
)
model = transformers.AutoModelForCausalLM.from_pretrained(
    model_id,
    trust_remote_code=True,
    config=model_config,
    quantization_config=bnb_config,
    device_map='auto',
)
tokenizer = AutoTokenizer.from_pretrained(model_id)
time_2 = time()
print(f"Prepare model, tokenizer: {round(time_2-time_1, 3)} sec.")

KeyError
Cell In[39], line 2
  1 time_1 = time()
  2 model_config = transformers.AutoConfig.from_pretrained(
  3     model_id
  4 )
  5 model = transformers.AutoModelForCausalLM.from_pretrained(
  6     model_id,
  7     trust_remote_code=True,
  (...)
```

Figure 26: phi-2 with RAG results

**Discussion:** Our intention was to generate more contextual summaries with more accurate semantics and readability. But, as we failed to form the vector database from our attempt. On the other hand, there were API issues there. We generated API keys from the OpenAI server but those keys couldn't work for our model. However, the hugging face model is restricted to use in kaggle notebook or colab. Several approaches could not solve those issues. So, we couldn't generate a summary from the RAG approach in microsoft/phi-2. We explored multiple articles related to this but got the idea we can do it if we get enough resources and if it is possible to do this locally.

## Zephyr

**Investigation/Experiment:** We failed to generate with RAG approach with microsoft/phi-2, so our supervisor suggest us another LLM model HuggingFaceH4/zephyr-7b-beta which is larger than the microsoft/phi-2 model and optimized for summarization where microsoft/phi-2 was optimized for question-answering. We tried to load Zephyr Model several times but couldn't load the model to generate summaries as it requires higher memory (more than 29 GB) that we have in our kaggle

notebook. It is a 7 billion parameter model, so it requires higher memory space to run and we didn't have that much memory to run it.

**Result and Analysis:** Here, we can't manage the model to load in our kaggle notebook. Again, we tried to load the model in google colaboratory but the space in google colaboratory is lesser than the space in kaggle notebook. So, it also didn't run in colab. So, we couldn't generate anything valuable from this approach.

The screenshot shows a Jupyter Notebook interface with a red error bar at the top stating: "Your notebook tried to allocate more memory than is available. It has restarted." Below the error message, there is a code cell containing Python code for generating a summary. To the right of the code cell is a sidebar showing session statistics: Session 34m, Disk 51.6GB, CPU 128.00%, RAM 25.1GB, GPU 0.00%, GPU Memory 0 Bytes, and another GPU section showing 0.00% and 0 Bytes. At the bottom of the sidebar, there is a progress bar for loading checkpoint shards at 88% completion.

Figure 27: Zephyr results

**Discussion:** Our goal of this approach was to generate a better summary than using prompt engineering in the microsoft/phi-2 model. But, we couldn't generate summaries from this approach. If we had higher resources to use HuggingFaceH4/zephyr-7b-beta model, we could have generated summaries and compared those summaries with the summaries generated from microsoft/phi-2 model.

### Rouge score and Bleu score of phi-2

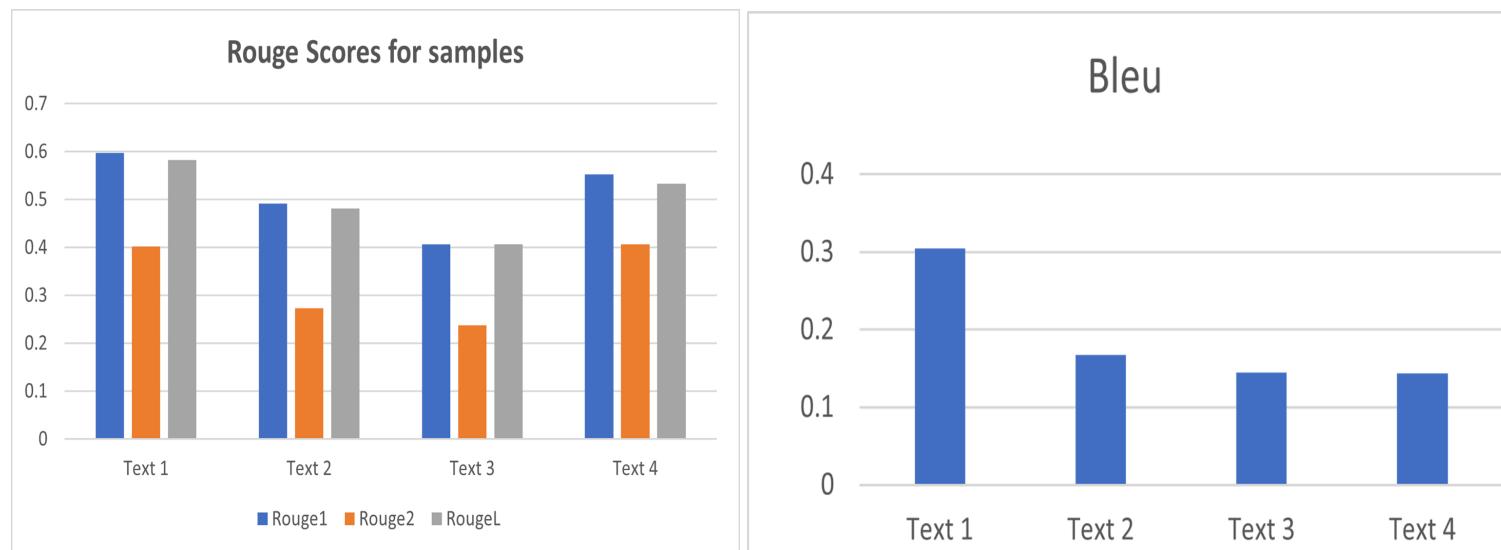


Figure 28: Rouge and Bleu score

The bar chart shows the rouge score and BLEU score for some samples. As we don't have the summary data in CNN dataset we used openAI summary as ground truth and compared to the output of Microsoft/phi-2 using prompt engineering. We can see the bleu score and rouge scores for some sample text.

## Comparison:

Category	Text	Open AI	Phi 2	grade score
Travel	<p>Ever noticed how plane seats appear to be getting smaller and smaller? With increasing numbers of people taking to the skies, some experts are questioning if having such packed out planes is putting passengers at risk. They say that the shrinking space on aeroplanes is not only uncomfortable - it's putting our health and safety in danger. More than squabbling over the armrest, shrinking space on planes putting our health and safety in danger? This week, a U.S consumer advisory group set up by the Department of Transportation said at a public hearing that while the government is happy to set standards for animals flying on planes, it doesn't stipulate a minimum amount of space for humans. 'In a world where animals have more rights to space and food than humans,' said Charlie Leocha, consumer representative on the committee. 'It is time that the DOT and FAA take a stand for humane treatment of passengers.' But could crowding on planes lead to more serious issues than fighting for space in the overhead lockers, crashing elbows and seat back kicking? Tests conducted by the FAA use planes with a 31 inch pitch, a standard which on some airlines has decreased. Many economy seats on United Airlines have 30 inches of room, while some airlines offer as little as 28 inches. Cynthia Corbett, a human factors researcher with the Federal Aviation Administration, conducts tests on how quickly passengers can leave a plane. But these tests are conducted using planes with 31 inches between each row of seats, a standard which on some airlines has decreased, reported the Detroit News. The distance between two seats from one point on a seat to the same point on the seat behind it is known as the pitch. While most airlines stick to a pitch of 31 inches or above, some fall below this. While United Airlines has 30 inches of space, Gulf Air economy seats have between 29 and 32 inches, Air Asia offers 29 inches and Spirit Airlines offers just 28 inches. British Airways has a seat pitch of 31 inches, while easyJet has 29 inches, Thomson's short haul seat pitch is 28 inches, and Virgin Atlantic's is 30-31.</p>	<p>The article discusses concerns over shrinking space in airplane cabins and its potential impact on passenger health and safety. Experts argue that the decreasing space not only makes travel uncomfortable but also poses risks to passengers. While government regulations address space standards for animals on flights, there are no meanders. This issue was raised by a U.S consumer advisory group, highlighting the need for humane treatment of passengers. Tests conducted by the FAA typically use planes with a 31-inch pitch, but many airlines offer less space, with some providing as little as 28 inches. The article concludes by listing various airlines and their seat pitch measurements, emphasizing the variability in space among carriers.minimum requirements for human pass</p>	<p>The text discusses how plane seats are getting smaller and how this affects passengers' health and safety. It mentions that the government does not set a minimum space for humans on planes, unlike for animals. It also compares the seat pitches of different airlines and how they vary from the standard of 31 inches. [T=0.3]</p>	<p>Hemingway: Open AI: 14 (poor) Phi 2: 10 (ok)</p> <p>Grammarly: OpenAI: 83/100 Phi 2: 93/100</p>
Science	<p>Cancer is a broad term. It describes Trusted Source, the disease that results when cellular changes cause the uncontrolled growth and division of cells. Some types of cancer cause rapid cell growth, while others cause cells to grow and divide at a slower rate. Certain forms of cancer result in visible growths called tumours, while others, such as leukaemia, do not. Most of the body's cells have specific functions and fixed lifespans. Cell death is part of a natural and beneficial phenomenon, which healthcare professionals call apoptosis. A cell receives instructions to die so that the body can replace it with a newer cell that functions better. Cancerous cells lack the components that instruct them to stop dividing and to die. As a result, they build up in the body, using oxygen and nutrients that would usually nourish other cells. Cancerous cells can form tumours, impair the immune system and cause other changes that prevent the body from functioning regularly. Cancerous cells may appear Trusted Source in one area, then spread via the lymph nodes. These are clusters of immune cells located throughout the body. There are many causes of cancer, and some are preventable. Preventable risk factors for cancer include Trusted Source: smoking, heavy alcohol consumption, excess body weight, physical inactivity, poor nutrition &amp; human papillomavirus infection. Other risk factors for cancer are not preventable. Currently, the most significant unpreventable risk factor is age. According to the American Cancer Society (ACS), doctors in the United States diagnose 88% Trusted Source of cancer cases in people ages 50 years or older. Genetic factors can contribute to the development of cancer. A person's genetic code tells their cells when to divide and expire. Changes in the genes can lead to faulty instructions, and cancer can result. Genes also influence the cells' production of proteins, and proteins carry many of the instructions for cellular growth and division. Some genes change proteins that would usually repair damaged cells. This can lead Trusted Source to a predisposition for cancer. If a parent has these genes, they may pass on the altered instructions to their offspring. A doctor may refer to this as an inherited gene mutation. These mutations may contribute to the development of up to 10% Trusted Source of cancer cases. Some genetic mutations that increase the risk of developing cancer occur after birth. Healthcare professionals refer to these changes as "acquired gene mutations". Possible causes include smoking and sun exposure. These genetic changes cause cancer more commonly than inherited gene mutations.</p>	<p>Cancer is a disease characterised by uncontrolled cell growth and division. It can manifest in various forms, including tumours or leukaemia. Cancerous cells lack the ability to stop dividing and die, leading to their accumulation in the body. Risk factors for cancer include smoking, alcohol consumption, obesity, and genetic factors. Some mutations in genes can increase the risk of cancer, either inherited or acquired later in life due to environmental factors like smoking or sun exposure.</p>	<p>Cancer is a disease that happens when cells grow and divide uncontrollably due to faulty instructions from the genetic code. Some types of cancer cause tumours, while others do not. Cancerous cells can spread throughout the body and impair the immune system. The most common cause of cancer is age, but some risk factors, such as smoking and heavy alcohol consumption, are preventable. Genetic mutations can also increase the risk of cancer, either inherited or acquired after birth. [T=0.3]</p>	<p>Hemingway: Open AI: 11 (ok) Phi 2: 10 (ok)</p> <p>OpenAI: 95/100 Phi 2: 100/100</p>

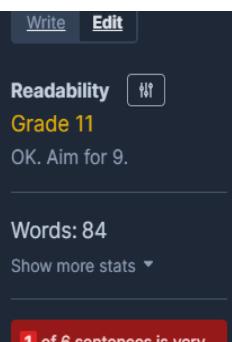
Figure 29: OpenAI vs Phi-2 comparison with hemingway vs grammarly score

## One sample for grammarly vs hemingway:

### Hemingway (OpenAI vs Phi-2)

#### Hemingway App makes your writing concise and correct.

Cancer is a disease characterised by uncontrolled cell growth and division. It can manifest in various forms, including tumours or leukaemia. Cancerous cells lack the ability to stop dividing and die, leading to their accumulation in the body. Risk factors for cancer include smoking, alcohol consumption, obesity, and genetic factors. Some mutations in genes can increase the risk of cancer, either inherited or acquired later in life due to environmental factors like smoking or sun exposure.



#### Hemingway App makes your writing concise and correct.

Cancer is a disease that happens when cells grow and divide uncontrollably due to faulty instructions from the genetic code. Some types of cancer cause tumours, while others do not. Cancerous cells can spread throughout the body and impair the immune system. The most common cause of cancer is age, but some risk factors, such as smoking and heavy alcohol consumption, are preventable. Genetic mutations can also increase the risk of cancer, either inherited or acquired after birth.

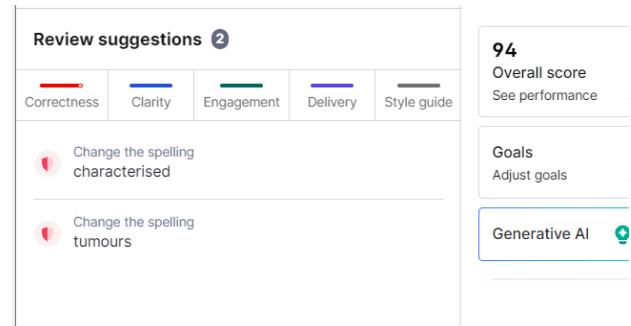


Figure 30: Hemingway score

## Grammarly

### For OpenAI Summary

Cancer is a disease characterised by uncontrolled cell growth and division. It can manifest in various forms, including tumours or leukemia. Cancerous cells cannot stop dividing and die, leading to their accumulation in the body. Risk factors for cancer include smoking, alcohol consumption, obesity, and genetic factors. Some mutations in genes can increase the risk of cancer, either inherited or acquired later in life, due to environmental factors like smoking or sun exposure.



### For Phi-2 Summary

Cancer is a disease that happens when cells uncontrollably due to faulty instructions from the body. Different types of cancer cause tumors, while others spread throughout the body and impair the normal functioning of organs. The common cause of cancer is age, but some factors, such as smoking and heavy alcohol consumption, are preventable. These factors also increase the risk of cancer, either inherited or acquired later in life.

## Performance

Great job! Your text scores 100 out of 100. This score represents the quality of writing in this document.



## Word count

Characters	482	Reading time	18 sec
Words	78	Speaking time	36 sec
Sentences	5		

## Readability

Metrics compared to other Grammarly users

Word length	5.1	Above average
Sentence length	15.6	Above average
Readability score	54 ⓘ	

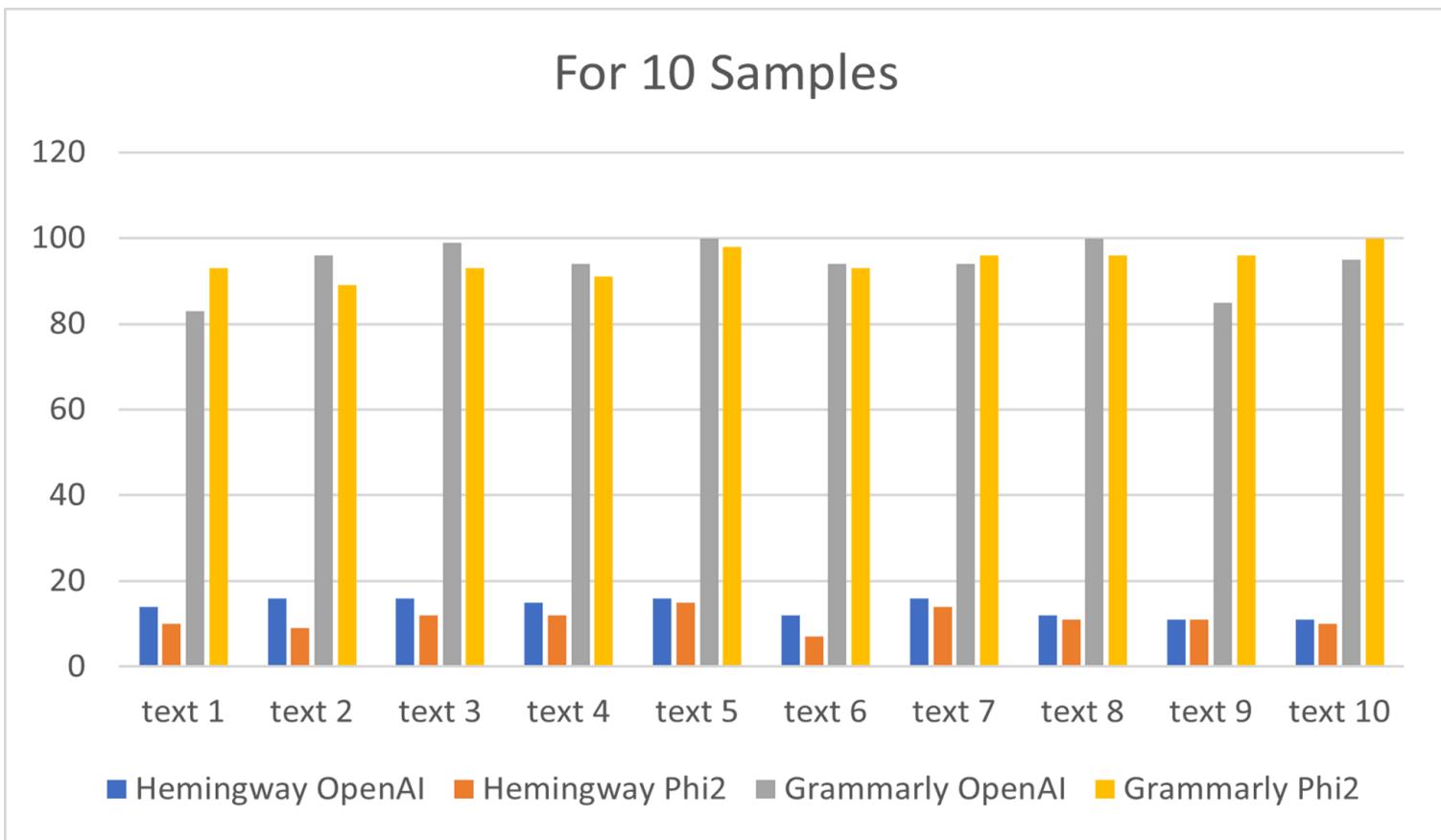
Figure 31: Grammarly score

**Main Text length:** 417 words

**Phi-2 summarized length:** 78 words

**Reduced Length:** 81%

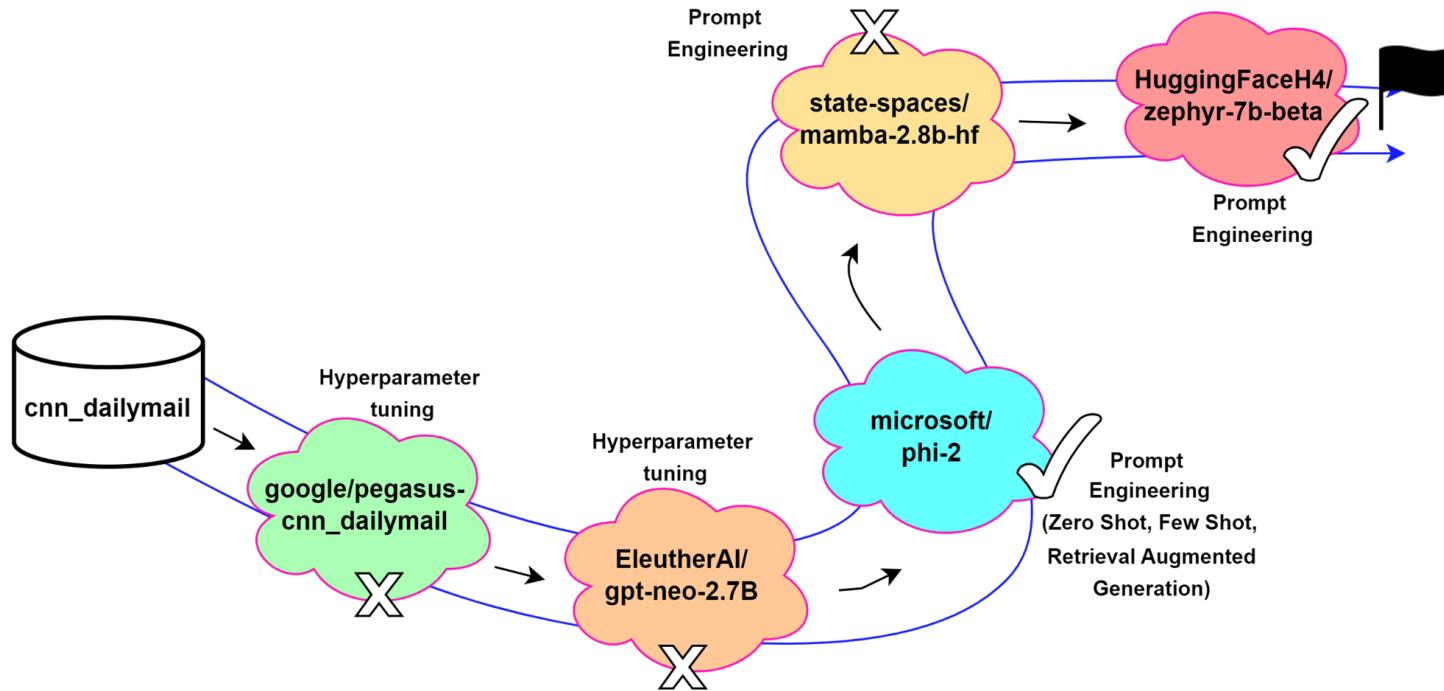
So, this is for one sample text compared with the grammarly and hemingway premium website. We tried it for at least 30 input texts. Most of the cases, Phi-2 gives a good score with reduced summary. In fact, it reduced the length very much. This indicates that our Small Language Model works absolutely fine with prompt engineering techniques with phi-2 better than other models.



*Figure 32: Hemingway and Grammarly score for 10 samples*

We evaluated the accuracy of the generated summary using Rouge and Bleu scores. Since we did not have the actual ground truth of the texts, we relied on the Hemingway score and Grammarly readability score to assess the generated texts from the Microsoft Phi-2 model and the OpenAI ChatGPT. Upon analysis, we found that the scores for the summaries generated from the Phi-2 model and ChatGPT are quite similar. This indicates we can achieve a proper contextual summary using a low-resource model like Microsoft Phi 2.

## Methodology Roadmap



*Figure 33: All models (methodology) Roadmap*

We started working on the CNN dataset on the Pegasus model, but after finding its drawback that it can't handle input script longer than 1000 words, we moved on to a new model named EleutherNetAI/GPT. We found 2 version of this model, one with 125M parameter and another one with 2.7B parameter. At first, we tried to generate summary with 2.7b version but soon realized we don't have that much resource to load that with happytransformer. So, we loaded this model 125M version using happy transformer, but we couldn't be able to generate the summary from it of its hyperparameter issue. After that, we used a new model named Microsoft/phi-2 and using it, we can generate a good level of summaries compared to openAI summaries for long text. We used prompt engineering to use the phi-2 model. But, to get more accurate summaries, we used another model, named Zephyr, which uses custom instruction to generate better output.

# Chapter 5 Impacts of the Project

## 5.1 Impact of this project on societal, health, safety, legal, and cultural issues

### Societal Impact

Using technology to summarize transcripts will significantly affect society. Instead of doing lengthy text, reading short ones with most of the information seems to be the most practical thing to do. Our project helps people save time and work better together by summarizing long texts more effectively.

**Efficiency and Productivity:** The project can enhance efficiency by reducing the time spent on reviewing long meeting recordings or notes. This, in turn, can improve productivity in both professional and personal settings.

**Accessibility:** Providing summarized and highlighted meeting content it can make the information more accessible to individuals with disabilities who may struggle with traditional note-taking methods.

### Health Impact

**Reduced Cognitive Load:** Automated summarization can help reduce cognitive load, preventing mental fatigue and burnout from information overload. This can contribute to better mental health for individuals who frequently attend meetings.

**Stress Reduction:** The project may reduce stress by simplifying the process of gathering and retaining important meeting insights.

### Safety Impact

**Compliance and Documentation:** It can contribute to regulatory compliance by ensuring that meeting discussions are accurately recorded and documented, which is crucial in fields like healthcare, finance, and law.

**Data Security:** There are potential safety concerns related to the security and privacy of sensitive meeting content, which must be addressed to prevent unauthorized access and data breaches.

### Legal Impact

**Evidentiary Value:** Automated meeting summaries can serve as valuable evidence in legal proceedings, providing a clear record of what transpired during a meeting.

**Intellectual Property:** There may be legal considerations surrounding the ownership and protection of the AI-generated meeting summaries and intellectual property rights.

**Copyright:** Meeting content may include copyrighted materials. It should consider copyright laws and fair use principles when summarizing and storing meeting data. Seek proper permissions or licenses when necessary.

### Cultural Impact

**Language and Localization:** The project can be tailored to support multiple languages and dialects, promoting cross-cultural collaboration and understanding.

**Cultural Sensitivity:** Care must be taken to ensure that the summarization and highlighting process does not inadvertently favor one cultural perspective or bias and that it respects cultural nuances

## 5.2 Impact of this project on Environment and Sustainability

### Environmental Impact

#### **Carbon Footprint**

The summary from a meeting session will be uploaded to cloud storage. So the paper usage will be less. That's why the carbon footprint will be less. Generating summaries from online meetings or classes, the carbon footprint is related to the environmental impact of the technology and infrastructure required to support the project.

- **Server Infrastructure:** The project involves processing video content to generate text summaries and highlighting points. This processing likely takes place on servers or data centers, which require electricity to operate. The carbon footprint is associated with the energy consumption of these servers. Using energy-efficient servers and data centers can help reduce this impact.
- **Video Streaming:** Online meetings and classes involve streaming video content to participants. The carbon footprint includes the energy required for video encoding, transmission, and decoding. Efficient video compression algorithms and content delivery networks (CDNs) can minimize the energy consumption associated with video streaming.
- **Softcopy:** After summarizing the meeting it will be printed as a softcopy. As it can be noted through only so in maximum cases hard copy use is less. As a result, soft copy can reduce the use of papers. So that it reduces the carbon emission and carbon footprint lower for paper production and ink.
- **User Devices:** Participants in online meetings or classes use various devices, such as computers, tablets, and smartphones, to access the content. The carbon footprint also depends on the energy efficiency of these devices. Encouraging users to use energy-efficient hardware can indirectly contribute to a lower carbon footprint.
- **Data Storage:** Storing video recordings of meetings or classes and their corresponding summaries can also have an environmental impact. This includes the energy consumption of data storage infrastructure. Employing data storage solutions that prioritize energy efficiency and data center sustainability can mitigate this impact.
- **Internet Connectivity:** Accessing online meetings and uploading video content for processing requires internet connectivity. The carbon footprint includes the energy consumption of network infrastructure and data transmission. The efficiency of internet service providers and the choice of network technologies can affect this footprint.

#### **Power consumption**

The summarization which involves generating summaries from online meetings or classes through video analysis, power consumption is a critical aspect related to the energy requirements of the technology and infrastructure involved. The system converted into the software version will not affect the power usage of a PC. So, the system will not affect power consumption very much.

- **Video Processing:** One of the primary contributors to power consumption is the video processing phase. Analyzing video content, extracting key information, and generating text summaries and highlights demand computational resources. The power consumption is directly related to the computing power and efficiency of the hardware used for these tasks.
- **Server Infrastructure:** The servers or data centers that host the video analysis algorithms and services require a significant amount of power to operate. The more powerful and numerous the servers, the higher the overall power consumption.
- **Data Transmission:** When video content is uploaded to a platform and summaries are downloaded or accessed by users, data transmission occurs. This involves power consumption in network equipment, including routers, switches, and data centers. The efficiency of data transmission can impact power consumption.
- **Electricity:** It powers data centers, servers, networking equipment, and user devices used for video analysis and summarization. The greenhouse effect refers to the trapping of heat in the Earth's atmosphere due to the accumulation of greenhouse gasses, primarily carbon dioxide (CO<sub>2</sub>). Power consumption, particularly if it relies on electricity generated from fossil fuels, can contribute to increased CO<sub>2</sub> emissions. These emissions intensify the greenhouse effect, leading to global warming and climate change.
- **User Devices:** Power consumption also extends to the devices used by meeting or class participants. This includes computers, smartphones, tablets, and other devices that access and interact with the platform. Efficient device usage can help reduce the power consumption associated with your project.
- **Real-time Processing:** If this aims to provide real-time summarization and highlighting during live meetings or classes, it may require specialized hardware, such as GPUs (Graphics Processing Units) or TPUs (Tensor Processing Units), which consume varying amounts of power depending on their efficiency and usage.

## Energy Efficiency

Hosting virtual meetings typically consumes less energy compared to in-person meetings, especially when considering factors like heating, cooling, and lighting of physical meeting spaces.

- **Reduced Environmental Impact:** Energy efficiency is crucial for minimizing the environmental footprint of the meeting summarization. By using less electricity and computational resources, it can contribute to lower carbon emissions and reduce its overall impact on the environment, helping combat climate change.
- **Physical Meeting vs Online Meeting:** Energy efficiency, comparing physical meetings to online meetings reveals important differences. Physical meetings typically involve travel, which consumes energy and emits carbon emissions, while online meetings require less energy for commuting. However, online meetings rely on electricity to power devices and data centers, introducing their own energy consumption concerns. The energy efficiency of online meetings depends on factors such as server infrastructure, data transmission, and device usage. Optimizing these elements can make online meetings more energy-efficient than physical ones, particularly when considering reduced travel-related emissions. Nonetheless, balancing energy efficiency in both meeting formats is crucial for achieving sustainability goals.
- **Efficient use of Internet and Electricity:** Efficient use of the Internet and electricity is essential for maximizing energy efficiency. Regarding the Internet, using data compression techniques, content delivery networks (CDNs), and efficient data transmission protocols can reduce energy consumption during data transfer. This minimizes the environmental impact of

online meetings and video analysis. Additionally, optimizing server infrastructure and algorithms can lower electricity consumption during video processing.

- **Cost Savings:** Improving energy efficiency can lead to significant cost savings. Lower power consumption means reduced energy bills, which can be especially important for data center operations and large-scale computational tasks.
- **Enhanced Scalability:** Energy-efficient algorithms and hardware can scale more effectively. As this grows, efficient resource utilization ensures that it remains cost-effective and environmentally responsible.
- **Related Energy:** Online meetings promote energy efficiency by reducing the carbon emissions and energy consumption associated with transportation to physical meetings. Additionally, they lead to less resource-intensive food production and waste, further enhancing their environmental benefits.

Finally, emphasizing energy efficiency throughout the online meeting summary is key to minimizing environmental impact, reducing costs, and aligning with sustainability goals, making it both effective and environmentally responsible.

## **Environmental Impact of Computation:**

**High Energy Consumption:** Running complex algorithms and training large NLP models requires significant processing power. This translates to massive electricity demands, often met by sources like fossil fuels, leading to greenhouse gas emissions and contributing to climate change.

**E-waste Generation:** The continuous production, use, and disposal of electronic devices associated with computation lead to electronic waste (e-waste). Improper e-waste management pollutes landfills with toxic materials and poses health risks.

**Resource Depletion:** Mining the raw materials required for computer components can lead to environmental degradation through deforestation, water pollution, and soil contamination.

## **Some positive sides:**

**Reduced Paper Consumption:** By automatically summarizing and highlighting key points from meetings, it eliminates the need for physical notes and reports. This translates to a substantial reduction in paper usage, saving trees and minimizing deforestation.

**Increased Efficiency:** This promotes a paperless workflow, fostering a more sustainable work environment. Easy access to condensed meeting information saves time and resources compared to manually reviewing lengthy recordings or notes.

**Improved Decision Making:** By effectively summarizing discussions and highlighting crucial points, the project ensures better understanding of meeting content. This can lead to well-informed choices based on the captured information, potentially promoting sustainable practices within the organization.

## **Sustainability**

### **In long run**

- **Resource Efficient Decision Making:** By efficiently reducing meeting transcripts and other textual data, it reduces the time and effort required for manual note-taking. Because of this optimization, firms and organizations can allocate human and computing resources more effectively, increasing productivity and resource efficiency.

- **Resource Efficiency and Long-term Impact:** Our project allows it to enable it to efficiently handle growing computational demands and technological advancements. This ensures that the project can continue to meet the needs of users and organizations as they evolve over time, leading to increased productivity and effectiveness. By leveraging scalable NLP models and computing infrastructure, our project is future-proofed against changes in technology and data requirements. This ensures that the project remains relevant and sustainable in the long run, capable of adapting to emerging trends and advancements in the field of natural language processing.
- **User-Centric Design:** Prioritize user experience and enhance the project's features and functionalities. By understanding and addressing user needs, the project can maintain relevance and user engagement over time.
- **Scalable Infrastructure:** Invest in scalable computing infrastructure that can accommodate increasing data volumes and user demands as the project grows. This ensures consistent performance and reliability, even as the project expands.
- **Continuous Innovation:** Foster a culture of innovation within the project team to drive ongoing improvements and stay ahead of technological advancements. Regularly explore new ideas, techniques, and technologies to maintain a competitive edge in the field.
- **Community Engagement:** Build a strong community around the project by engaging with users, contributors, and stakeholders. Encourage collaboration, knowledge sharing, and investment in the project's success.
- **Robust Documentation and Support:** Provide comprehensive documentation and support resources to facilitate user onboarding, troubleshooting, and continued usage. Clear, accessible documentation helps users navigate the project effectively and reduces barriers to adoption.
- **Educational Outreach:** Offer educational resources, training programs, and workshops to empower users and stakeholders with the knowledge and skills needed to maximize the project's value.
- **Long-Term Vision:** Maintain a clear long-term vision and roadmap for the project's evolution and growth. Set ambitious yet achievable goals and milestones to guide strategic decision-making and maintain momentum toward sustained success.

### **Effect on the mass use:**

- **Technological Dependence:** Mass reliance on automated summarization technologies could lead to dependency on digital tools and platforms. Organizations may become overly reliant on our project's capabilities, leading to potential disruptions if there are issues with the technology or infrastructure supporting it.
- **Digital Divide:** The widespread adoption of our project may exacerbate existing disparities in access to technology and digital literacy skills. Organizations or individuals lacking access to digital resources or adequate training may be left behind, widening the digital divide and hindering inclusivity.
- **Privacy Concerns:** Mass use of automated summarization technologies raises privacy concerns related to the handling and storage of sensitive information. Organizations must implement robust security measures to safeguard data privacy and comply with regulations, addressing potential risks associated with unauthorized access or data breaches.

# Chapter 6 Project Planning

Task Name	Summer 2023				Intersession 2023		Spring 2024			
	Aug-23	Sep-23	Oct-23	Nov-23	Dec-23	Jan-24	Feb-24	Mar-24	Apr-24	May-24
Prepare plan	■									
Research Models & Datasets	■	■								
Problem Finding	■									
Submit Project Proposal		■								
Progress Report 1		■	■							
System Design		■								
Result Analysis		■								
Progress Report 2		■	■							
Model Training			■	■						
Testing & Evaluation			■	■						
Result Analysis				■	■					
Research New Model				■						
Present Summer-2023 Progress					■					
Build System						■				
Model Training						■	■			
Testing & Evaluation						■	■			
Model Analysis & Compare								■	■	■
Prompt Engineering								■	■	■
RAG								■	■	
Final Evaluation									■	■
Project Demonstration									■	
Final Presentation										■
Final Report										■

**Figure 34: Project Planning Gantt chart**

# Chapter 7 Conclusions

## 7.1 Summary

In conclusion, Our project targets a broad audience by providing accessible summarization tools. NLP meeting summarization automates the process of condensing discussions into concise summaries, saving time and making content more accessible. Our findings not only enhance our current offerings but also provide valuable insights for future Researchers. The model aims to generate a better contextual summary from news or long transcripts using a SLM model. It involves a thorough process of model evaluation and optimization. Initially, multiple models, including Pegasus-CNN-DailyMail and EleutherAI/gpt-neo-125M, were analyzed to determine their effectiveness in summary generation. Among these, we found a better model - the Microsoft Phi-2 model, a small language model (SLM), as the best performer. To enhance the quality of the generated summaries, we utilized various prompt engineering techniques such as zero-shot, few-shot, and Retrieval-Augmented Generation (RAG). The implementation of RAG (Retrieval-Augmented Generation) makes the summaries much better by adding more context and retrieving useful information. Furthermore, using low-cost fine-tuning and prompt engineering techniques, this model can preserve the semantics and meaning of the textual context properly.

## 7.2 Limitations

All our experiments and models have at least some issues with memory. Even any Large Language Model (LLM) has these major restrictions. Already mentioned with such a big dataset called cnn-news dataset that size is almost 9GB. In fact, with this dataset our LLM model could not be able to work locally. In Locally, our memory can not load this. It at least needed 32GB external memory or multiple GPUs. As a result, it is quite impossible to go with this. LLM models need big resources and enough cost. For those drawbacks, we worked with the Small language Model (SLM) which worked so well. Therefore, it can be said that if we have enough resources and gpu we can easily generate the best summary within local devices.

### *Explanation:*

- **Resource Limitation (VRAM, GPU, Disk Space):** Basically, resource limitation includes limitations in VRAM (Video Random Access Memory), GPU (Graphics Processing Unit), and disk space. Such limitations can impact the project's ability to process and analyze data efficiently. For example, during execution, the model demanded more resources than our available GPU could store, leading to memory issues. To address this, we had to reduce the epoch size.
- **Library Support Issues (Some libraries needed to be upgraded):** Library support issues arise when the software libraries or dependencies used in projects require upgrades. This can lead to compatibility problems and affect the smooth execution of the project. Upgrading libraries is essential to keep the project up to date but can also be time-consuming and may introduce new issues.

- **Model Selection:** Model selection challenges revolve around choosing the most suitable NLP model for the task. Sometimes, models may struggle to understand the meaning of specific inputs or contexts. This can result in incorrect outputs. For example, Our main challenge in this project was selecting an appropriate model. Firstly, we used the model “Google pegasus cnn daily mail.” The initial choice worked well for shorter texts but struggled with longer transcripts, resulting in unsuitable summaries. To address this, we experimented with segmenting long transcripts, but in that case, it led to a loss of context. So, we canceled the model and started work with the new model, “ElutherAI/get-neo-2.7 B”. Then again shift with Phi-2 and Prompt Engineering method with mamba testing.
- **Coding Errors During Fine-Tuning:** Coding errors during the fine-tuning phase are common challenges. These errors can include bugs in the code, misconfigurations, or hyperparameter issues. In our case, fine-tuning was a new experience, and we encountered multiple issues that required adjustments and changes as we refined our approach.
- **Network Connectivity Issues (Colab):** Network connectivity issues are related to interruptions or instability in internet connections, mainly when using cloud-based platforms like Google Colab.
- **Power Stability Issues:** Power stability issues involve disruptions caused by unstable power sources. These issues can lead to unexpected shutdowns and data loss.

### 7.3 Future Improvement

In the future, We plan for local deployment and API integration, allowing flexible use across different platforms. While barriers in long text summarization, such as maintaining context in large discussion scripts with multiple contexts, and model limitations where scripts longer than 1200 words may cause breakdowns. We continuously worked on overcoming these issues to deliver accurate and coherent summaries. As we expand, our commitment to innovation and improvement ensures that we remain at the leading edge of summarization technology. It is a valuable tool for teams of all sizes, helping them to save time, improve productivity, and make better decisions.

- **Business Model:** So far, our project has been implemented on various models for summarization from a transcript. Our business model involves deploying an unpaid version - basic summarization is free with limited customization and integration. News reporters can understand the news through generated summaries. Users can upgrade for more options and support and for the paid version - Users have to get a subscription for extra features like customizing, tuning, and better output summaries, advanced algorithms for better accuracy, can use locally, integration with popular tools, and priority support. As we grow, we'll expand our services and keep improving to stay ahead.
- **Advanced Models:** Explore and integrate state-of-the-art models specialized in handling longer transcripts or diverse document types for enhanced summarization. Initially, we will start to work with the new model "ElutherAI/get-neo-2.7 B."
- **Hybrid Approaches:** Investigate hybrid summarization approaches that blend extractive and abstractive methods, aiming to maintain contextual understanding while condensing information effectively.
- **Domain-Specific Models:** Develop domain-specific models trained on meeting conversations or lecture notes to capture the essence of such content precisely.

- **User-Centric Focus:** We will gather user feedback to tailor the summarization process to meet their needs and preferences, refining the models accordingly.
- **Efficiency Enhancement:** We will focus on optimizing computational resources and fine-tuning algorithms to reduce processing time without compromising the quality of summaries.
- **Collaborative Research:** Collaborate with academic or industry experts to incorporate cutting-edge advancements in the field of NLP and summarization into the project.

In conclusion, Our project targets a broad audience by providing accessible summarization tools. Our findings not only enhance our current offerings but also provide valuable insights for future Researchers. In the future, We plan for local deployment and API integration, allowing flexible use across different platforms. While barriers in long text summarization, such as maintaining context in large discussion scripts with multiple contexts, and model limitations where scripts longer than 1200 words may cause breakdowns. We continuously worked on overcoming these issues to deliver accurate and coherent summaries. As we expand, our commitment to innovation and improvement ensures that we remain at the leading edge of summarization technology.

## References

1. Saiyyad, M. M., & Patil, N. N. (2024). Text Summarization Using Deep Learning Techniques: A Review. MDPI. <https://doi.org/10.3390/engproc2023059194>
2. Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang\*, Jinghua Tan (2024). A Comprehensive Survey on Process-Oriented Automatic Text Summarization with Exploration of LLM-Based Methods. (n.d.). Ar5iv. <https://ar5iv.labs.arxiv.org/html/2403.02901>
3. Kumar, S., Solanki, A. An abstractive text summarization technique using transformer model with self-attention mechanism. *Neural Comput & Applic* 35, 18603–18622 (2023). <https://doi.org/10.1007/s00521-023-08687-7>
4. Basyal, L., & Sanghvi, M. (2023, October 16). Text summarization using large language models: a comparative study of MPT-7B-Instruct, Falcon-7b-Instruct, and OpenAI Chat-GPT models. arXiv.org. <https://arxiv.org/abs/2310.10449v2>
5. Zhang, T., Ladakh, F., Durmus, E., Liang, P., McKeown, K., & Hashimoto, T. B. (2023, January 31). Benchmarking large language models for news summarization. arXiv.org. <https://arxiv.org/abs/2301.13848>
6. Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024, February 5). A Systematic survey of prompt engineering in large language Models: Techniques and applications. arXiv.org. <https://arxiv.org/abs/2402.07927>
7. Zhong, M., Liu, Y., Xu, Y., Zhu, C., & Zeng, M. (2022). DialogLM: Pre-trained Model for Long Dialogue Understanding and Summarization. Proceedings of the AAAI Conference on Artificial Intelligence, 36(10), 11765- 11773. <https://doi.org/10.1609/aaai.v36i10.21432>
8. R. Boorugu and G. Ramesh, "A Survey on NLP-based Text Summarization for Summarizing Product Reviews," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 352-356, doi: 10.1109/ICIRCA48905.2020.91833555
9. Murray, Gabriel & Carenini, Giuseppe. (2008). Summarizing Spoken and Written Conversations.. 773-782. 10.3115/1613715.1613813
10. Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond T. Ng. 2013. Abstractive Meeting Summarization with Entailment and Fusion. In Proceedings of the 14th European Workshop on Natural Language Generation, pages 136–146, Sofia, Bulgaria. Association for Computational Linguistics. Abstractive Meeting Summarization with Entailment and Fusion (Mehdad et al., ENLG 2013)
11. Moiyadi, Hamza S., et al. "NLP Based Text Summarization Using Semantic Analysis." International Journal of Advanced Engineering, Management and Science, vol. 2, no. 10, Oct. 2016. RIS (Mendeley, Zotero, EndNote, RefWorks) BibTeX (LaTeX)
12. Chinatsu Aone, Mary Ellen Okurowski, and James Gorlinsky. 1998. Trainable, Scalable Summarization Using Robust NLP and Machine Learning. In 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1, pages 62–66, Montreal, Quebec, Canada. Association for Computational Linguistics.
13. Agrawal, Y., Thakre, A., Tapas, T., Kedia, A., Telkhade, Y., & Rathod, V. (2021). Comparative analysis of NLP models for Google Meet Transcript summarization. EasyChair Preprint,(5404)

14. OpenAI. (2023). ChatGPT [Large language model]. <https://chat.openai.com>
15. Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In Proceedings of the 37th International Conference on Machine Learning (pp. 11328-11339). PMLR. Available from <https://proceedings.mlr.press/v119/zhang20ae.html>.
16. What is Natural Language Processing? | IBM. (n.d.).  
<https://www.ibm.com/topics/natural-language-processing> (11)
17. Foster, K. (2023). Text Summarization for NLP: 5 Best APIs, AI Models, and AI Summarizers in 2023. News, Tutorials, AI Research.  
<https://www.assemblyai.com/blog/text-summarization-nlp-5-best-apis/>
18. What is Natural Language Processing? - NLP Explained - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/what-is/nlp/>
19. Wikipedia contributors. (2023). Natural language processing. Wikipedia.  
[https://en.wikipedia.org/wiki/Natural\\_language\\_processing](https://en.wikipedia.org/wiki/Natural_language_processing) V. Collison, J. (2023, May 2). Maximizing Productivity with Text Summarization: How Chat GPT Can Help You Save Time. The Average Guy Network.  
<https://theaverageguy.tv/2023/05/01/maximizing-productivity-with-text-summarization-how-chat-gpt-can-help-you-save-time/>
20. Singh, A. (2022, August 28). Natural language processing (NLP) - DataSciences.io. DataSciences.io -. <https://datasciences.io/course/natural-language-processingnlp>
21. *Natural language processing to extract social risk factors influencing health.* (2023, August 23). ScienceDaily. <https://www.sciencedaily.com/releases/2023/08/230821114409.htm>
22. Podder, S. (2020, September 18). *How green is your software?* Harvard Business Review.  
<https://hbr.org/2020/09/how-green-is-your-software>
23. Plata, M. (2023). Guide to reduce your software's carbon footprint. *Applover*.  
<https://applover.com/blog/guide-to-reduce-your-softwares-carbon-footprint/>
24. *google/pegasus-cnn\_dailymail · Hugging Face.* (n.d.).  
[https://huggingface.co/google/pegasus-cnn\\_dailymail](https://huggingface.co/google/pegasus-cnn_dailymail)