National College of Ireland

# Text Summarization using Pegasus Model

MSc Research Project
MSCAI1- Master of Science in Artificial Intelligence


Anu Benny
Student ID: x23104937

School of Computing
National College of Ireland


Supervisor: Prof. Mayank Jain

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Anu Benny |
| **Student ID:** | x23104937 |
| **Programme:** | MSCAI1 - Master of Science in Artificial Intelligence |
| **Year:** | 2024 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Prof. Mayank Jain |
| **Submission Due Date:** | 12/08/2024 |
| **Project Title:** | Text Summarization using Pegasus Model |
| **Word Count:** | 4300 |
| **Page Count:** | 13 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 12th August 2024 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | □ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Text Summarization using Pegasus Model

Anu Benny
x23104937

**Abstract**

As digital content grows at an unprecedented rate, dealing with information overload has become a vital concern. In order to improve summarization from conversational text, this project uses the SAMSum dataset to fine-tune the Pegasus model. Our goal was to strengthen the model's capacity to produce clear and succinct summaries, which led to a significant improvement in the Rouge score—a crucial indicator of summarization quality. Because of the vast amount of text data that is generated every day—including emails and chat logs—effective summarization is essential. The Rouge-1 score was successfully raised by our fine-tuning, indicating better summarization accuracy. To further highlight the model's adaptability, I increased its capacity to summarize YouTube videos and extract and summarize text from photos. These enhancements provide useful advantages for more effective summarization and are in line with recent developments in NLP. In order to get over current constraints and improve the model's wider applicability, future work will concentrate on better optimizing it for intricate, multimodal inputs.
**Keywords**: Summarization, finetune, Pegasus

# 1 Introduction

In today's environment, with the explosion of digital content, summarizing aids in information overload by allowing consumers to quickly absorb vital elements without having to read lengthy publications. According to IDC research, the amount of digital data is expected to surpass 175 zettabytes by 2025, highlighting the urgent need for efficient summarization tools Rydning et al. (2018). The relevance of summarizing in boosting productivity, promoting better decision-making, and increasing information retrieval across a range of businesses, including the news, healthcare, and legal sectors, is highlighted by this increase in data. Condensing a large text into a shorter version while maintaining its essential details and general meaning is known as text summarizing. Text summarizing helps extract key insights and saves time by simplifying the intake of large amounts of information. It is becoming increasingly important in our data-driven environment. Natural language processing (NLP) has advanced, yet producing summaries of high quality is still a difficult challenge. There are two types of text-summarizing techniques: extractive and abstractive Mittal et al. (2023). Extractive summarization, which chooses essential sentences from a source text, frequently fails to capture subtle context and meaning. Abstractive summarization, which generates new sentences to convey the core concepts, offers improved coherence and readability, but it is computationally intensive and requires sophisticated models.

Transformer-based models have transformed natural language processing (NLP) by adding a technique known as self-attention, which enables models to focus on many areas of a text at once. This architecture serves as the basis for Pegasus and is the engine behind

numerous sophisticated models, such as BERT, GPT, and T5. The Pegasus model, which stands for "Pre-training with Extracted Gap-Sentences for Abstractive Summarization Sequence-to-Sequence Models," is intended primarily for the abstractive summary of text. Pegasus has been optimized to produce clear, coherent summaries of lengthy texts, in contrast to other general-purpose models Etemad et al. (2021). A primary advancement of Pegasus is its pre-training approach, which mimics a summarizing assignment by masking full phrases instead of individual words. This enhances the model's capacity to provide summaries and helps it comprehend a document's overall context. While Pegasus excels in summarization jobs due to its unique pre-training, it has an advantage in capturing the substance of a material. ROUGE is a collection of measures typically used to assess the quality of summaries produced by models such as Pegasus. It is especially useful for evaluating summarization models since it determines how much of the key content in the reference summary is captured by the model.

In this study, the Pegasus model for text summarization has been optimized on a dialogue dataset to improve its summary performance. This increases the model's adaptability and efficiency in summarizing various kinds of content. The Pegasus model is fine-tuned by training it on the dataset so that it can conform to the unique traits and writing styles of the dataset. Following refinement, the model's performance is measured by computing the ROUGE score, a metric that compares the quality of the generated summaries to reference summaries to determine how well the model performed. This evaluation reveals how effectively the fine-tuned model works in creating accurate and coherent summaries of each dataset. In today's fast-paced digital world, organizing the vast volume of content available online is hard. Summarizing tools are useful in this situation and provide a lot of benefits in terms of productivity and efficiency. For example, summarizing YouTube videos helps users save time by condensing the main points of long videos into brief summaries. As a result, viewers may easily understand the main ideas without having to see the entire film, which facilitates decision-making by enabling them to determine whether the information is worth more attention. In the same way, image summarizing improves the simplicity of visual content interpretation.

Research Questions and objectives:

1) How useful is the Pegasus model for summarizing discussions, especially when fine-tuned with the SAMSum dataset?

2) What effects do the size and diversity of the SAMSum dataset have on the quality of the generated summaries and the process of fine-tuning?

The Pegasus model's pre-trained architecture, which is intended to generate summaries, makes it very adept for abstractive summarization. Pegasus may be trained to provide clear and logical dialogue summaries by utilizing the conversational data in the SAMSum dataset. Metrics such as ROUGE scores can be used to assess the model's performance; improvements made after fine-tuning would suggest a successful adaptation to the dialogue summary. The SAMSum dataset's size and diversity are important factors in fine-tuning. The model is able to generate summaries that are more broadly applicable and exhibit improved generalization when it is given access to a diversified dataset featuring a diversity of themes and discourse patterns. On the other hand, the model may overfit to particular patterns in the data if the dataset is too limited or lacks diversity, which would result in worse generalization and performance in hypothetical discussion

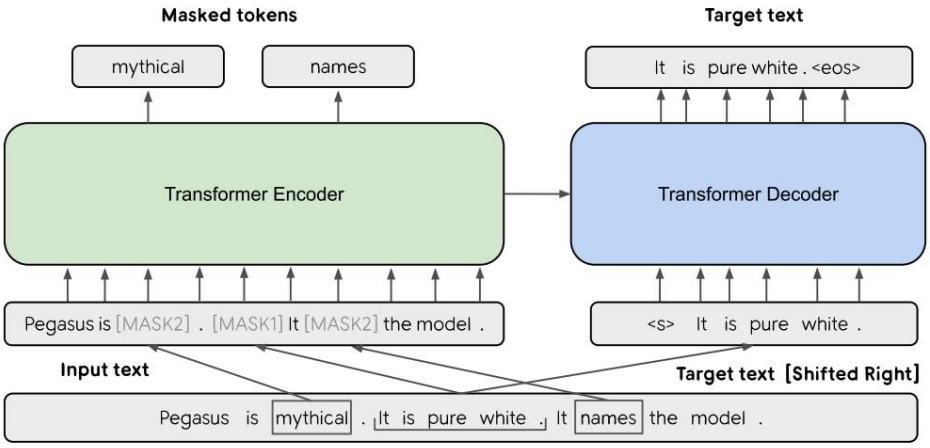


Figure 1: Architecture

scenarios.

My project uses the SAMSum dataset to fine-tune the Pegasus model's text-summarizing capabilities. Following fine-tuning, the ROUGE score—a statistic that contrasts the generated summaries with reference summaries—is used to assess the model's correctness. The project requires summarizing content from two sources: YouTube video transcriptions and text extracted from photos. The method is trained to summarize text information taken from images and to reduce YouTube video transcriptions into brief summaries. The model can effectively handle a variety of input data types due to its dual methodology. The research intends to improve the model's capacity to generate precise and cogent summaries, making it a flexible tool for processing various media forms by incorporating video and image-to-text summarizing.

# 2 Related Work

In today's environment, where information is being generated at an unprecedented rate, it is critical to rapidly understand and digest enormous amounts of information. This is made easier by text summarizing, which takes lengthy papers and distills them into concise, insightful summaries that hit all the important points. In addition to saving time, this facilitates access to and utilization of critical information by individuals and organizations without entangling them in pointless details. Effective summarizing tools enable people to stay informed and make smarter decisions more quickly, whether they are handling customer communications, reading news stories, or browsing through research papers. It is more crucial than ever to have tools that can effectively summarize information because text-based data is constantly increasing Sun et al. (2024).

An approach known as abstractive summarization only extracts and rearranges the most important lines from a text and generates brand-new sentences that more naturally and succinctly express the core concepts Asmitha et al. (2024). This is more similar to how individuals rephrase information while maintaining its original meaning when summarizing it. Advanced technologies, such as deep learning algorithms and powerful

neural networks, are needed to build efficient abstractive summarization models. These models are able to produce concise, clear, and pertinent summaries because they have been trained on a vast quantity of data to comprehend the context and meaning of words.

Natural language processing activities such as text summarization were commonly performed using Pegasus, BERT (Bidirectional Encoder Representations from Transformers), T5 (Text-To-Text Transfer Transformer), and BART (Bidirectional and Auto-Regressive Transformers). For example, BERT was not expressly made for text production; instead, it was mainly employed for jobs where it excelled in understanding context, such as question-answering and classification. However, T5 and BART were built to handle both understanding and producing text, they were more adaptable. In order to enhance the quality of the generated text, BART included parts of both encoder-decoder models, whereas T5 treated tasks as text-to-text transformations Shakil et al. (2024). While BART, the transformer-based summary model, is excellent at producing translations and paraphrases, it might not be as well-suited for abstractive summarization as Pegasus. BART employs a denoising autoencoder technique. Although Pegasus was particularly trained on summarizing datasets such as CNN/DailyMail, it often outperforms T5 on long-text summarization tasks. T5 is a versatile model that can handle a wide range of tasks, including summarization.

When Google developed Pegasus, it raised the standard for abstractive summarization. In contrast to earlier models, Pegasus specialized in pre-training, utilizing a cutting-edge method known as "Gap Sentences Generation" (GSG) in order to provide summaries. Using this technique, significant sentences in a document are masked, and the model is then trained to produce those sentences based on the surrounding content Zhang et al. (2020). This method works especially well for summarizing activities since it helps Pegasus have a stronger comprehension of how to simplify and rephrase material. Although other models, such as BART and T5, may also perform abstractive summarizing, Pegasus is able to generate summaries that are frequently more coherent and closely resemble the way humans summarize text due to its unique pre-training technique. Fine-tuning the Pegasus model is critical since it helps the model adapt to different types of content and domains, making it more efficient for certain summarization tasks. Pegasus learns the specifics of diverse text structures and styles by training on a variety of datasets, including news items, dialogues, and scientific papers. This results in summaries that are more accurate and pertinent to the context. By comparing the generated summaries to reference summaries and measuring overlaps in n-grams, word sequences, and word pairs, the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score provides a quantitative evaluation of the model's performance and helps us determine how accurate Pegasus is.

YouTube video summaries have major benefits in an era where video content is abundant and time is limited Panthagani et al. (2024). Users may rapidly understand the main themes without having to watch long content because of the brief summaries of videos that are provided, making it an invaluable tool for effective information intake. This is especially true for long talks, lectures, and instructional videos when the essential points can be condensed into succinct summaries. By extracting and compressing the transcript and concentrating on the most important information, the technology summarizes efficiently. But there are drawbacks to this strategy. The correctness of the transcript has a significant impact on the quality of the summary; during the summarization process, subtleties in tone or visual components may be lost Srivastava et

al. (2024). Notwithstanding these difficulties, video summarization is nevertheless a potent instrument for improving our interactions with online video material.

Image summarization is very useful in industries such as digital asset management, where vast amounts of photos must be examined and classified efficiently. Users can rapidly recognize the most crucial visual components, such as focal objects, scenes, or features, by summarizing photos instead of carefully examining each one. This improves decision-making in situations where visual data is plentiful, in addition to saving time. Still, there are several drawbacks to image-summarizing. Occasionally, the procedure may oversimplify intricate visuals, possibly omitting context or minute elements that are crucial for complete comprehension. Furthermore, the algorithms employed to summarize information may not always accurately capture the subtleties of each image, which affects how successful summarization is Mahalakshmi and Fatima (2022). Notwithstanding these difficulties, image summarizing is still a useful technique for streamlining our processing and interpretation of visual data.

# 3 Methodology

## 3.1 Finetuning Pegasus

### 3.1.1 Data Collection

To train and evaluate the pegasus model, I used the SAMSum dataset, which contain dialogues, and their summaries. The Hugging Face dataset library was used to load this datasets. The Samsum dataset contains the columns dialogue, summaries, and id.

```
Split lengths: [14732, 819, 818]
Features: ['id', 'dialogue', 'summary']
```

Figure 2: Dataset Split

```
Dialogue:
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 :)

Summary:
Eric and Rob are going to watch a stand-up on youtube.
```

Figure 3: Dataset

### 3.1.2 Data Preprocessing

Preparing the raw data for modeling requires data preprocessing.

**Tokenization:** The Hugging Face Transformers library's AutoTokenizer, which transforms text into input tokens that can be fed into the model, is used to tokenize the dialogues and summaries. To maintain uniformity throughout the dataset and avoid excessively lengthy sequences, the maximum length of tokens for articles is set to 1024, while the maximum length for summaries is set to 128.

**Batch Processing:** To effectively manage memory, the dataset is processed in smaller batches using the generate-batch-sized-chunks function. Smaller dataset segments are produced by this function, allowing for simultaneous processing.

**Mapping the Preprocessing Function:** The map method is used to apply the features function to the whole dataset. By doing this, it is ensured that every data point in the dataset is converted into the right format needed for training the model.

### 3.1.3 Feature Extraction

The retrieved features consist of:

- input-ids: These are the input articles' tokenized representations.
- attention-mask: In the input data, this mask is utilized to distinguish between genuine tokens and padding tokens.
- labels: During training, the method will learn to predict these tokenized representations of the target summary.

The tokenizer is used to extract these features, which are crucial for the sequence-tosequence modeling function that the PEGASUS model is intended to carry out.

### 3.1.4 Modelling

Hugging Face's Trainer is used to fine-tune the model on the Samsum dataset, making the training process easier.

**Model Initialization:** The PEGASUS model is relocated to the appropriate device (CPU or GPU) and loaded with pre-trained weights from the google/pegasus-cnn-dailymail checkpoint.

**Training Setup:** A set of hyperparameters, such as the number of epochs, batch sizes, warm-up steps, and weight decay, are setup for the training. The model's performance can only be maximized by using these parameters.

**a) Time intervals (num_train_epochs=1)** - Since the SAMSUM dataset is tiny, the model may be able to learn the patterns in the data with fewer epochs. Excessive epoch fine-tuning may result in overfitting. Therefore, if the epochs are set to 1, the model will collect the required data without overfitting.

**b) Batch Size (per device eval batch size = 1, per device train batch size = 1) -** Especially when employing GPUs with limited memory, maintaining a small batch size helps prevent running out of memory.

**c) Steps in Gradient Accumulation (gradient_accumulation_steps=16)** - Because of memory limitations, the batch size is modest; gradient accumulation helps to mimic a higher effective batch size. Gradient accumulation over 16 steps causes the optimizer to change its weights every 16 batches, simulating a batch size of 16, which improves training stability.

**d) Steps for Warming Up (warmup_steps=500)** - In order to ensure smoother training and avoid the model diverging too early in the training process, the model starts with lower learning rates and progressively raises them over the warmup phase. The model's weights are gradually adjusted to the new task using a warmup consisting of 500 steps.

**e) Loss of Weight (weight_decay = 0.01) -** By punishing big weights, weight decay is a regularization approach used to minimize overfitting. A modest value of 0.01 guarantees that the model performs well in terms of generalization on the validation set and avoids overfitting to the training set.

**Training:** The model, tokenizer, training arguments, data collator, and datasets for assessment and training are instantiated into the Trainer class. Using the Samsum dataset, the PEGASUS model is refined by invoking the train() method.

**Evaluation:** Using ROUGE metrics to compare the generated summaries with the reference summaries, the model's performance is assessed on the test dataset following training.

**Model Saving:** The optimized model and tokenizer are stored for later use.

## 3.2 Video Summarization

Using YouTubeTranscriptApi, obtain the transcript from the YouTube video. Then the preprocessing methods are putting all of the transcript text into one string and due to input size limitations, dividing the text into parts for processing. Then the Modelling involves using the pipeline function from the transformers library, load a pre-trained summarization model. And utilizing the model to produce summaries for every text section.

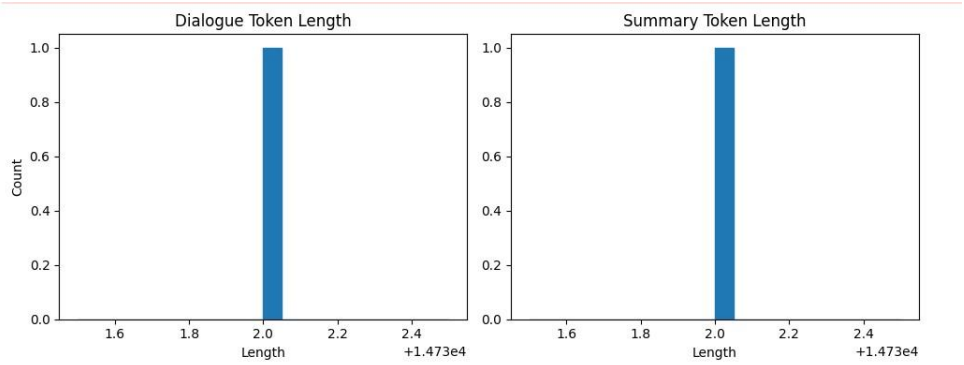


Figure 4: Histogram

### 3.3 Image Summarization

Loading the picture. Utilizing OCR to convert the image to text. Then taking textual information out of the picture. The text taken from the image is summarized using a pre-trained Pegasus model that has been loaded.

## 4 Design Specification

Initially, I built up the Python libraries required for managing model training and assessment, such as rouge-score, datasets, transformers, and others. Using the Transformers library, the code loads a pre-trained model (google/pegasus-cnn-dailymail) at the start. To handle massive volumes of data more efficiently, a dataset is divided into smaller batches using the chunks function. These batches are then used by the test-ds function to tokenize articles, produce summaries using the model, and then decode the summaries into legible text. To assess the performance of the model, these generated summaries are compared using ROUGE metrics with reference summaries. This batching strategy guarantees a methodical evaluation of the model's summarization skills while using less memory. The Samsum dataset was loaded through the Hugging Face datasets library. The dataset is divided into smaller pieces for processing in an effective manner after loading. It shows the feature names of the training data and prints the number of samples in each split. After that, a test example summary and the article are printed to help with data format comprehension. Subsequently, a summarization pipeline is constructed with the designated PEGASUS model checkpoint, producing a summary for a test article.

To assess the performance of the summarization, ROUGE metrics are loaded. The test dataset's Rouge scores are calculated using the test-ds function; the results are arranged into a dictionary and shown in a Pandas DataFrame. In order to show the distributions of token lengths for dialogues and summaries, the script additionally analyzes them and generates histograms. Using the provided tokenizer, the features method tokenizes the summaries and the articles. To maintain consistency, it establishes a maximum length for articles and summaries and truncates any material that goes beyond that limit. Inputids, attention-mask, and labels are the dictionaries that this method returns. These are

necessary for feeding the data into the model. The function is then applied to each sample in the dataset by processing it via the map method. In doing so, a format appropriate for training is created from the raw dataset. Subsequently, the DataCollatorForSeq2Seq is set up to manage dynamic sequence padding during training, ensuring that every batch is padded to the batch's longest sequence.

The configuration of the training process is done through the training arguments. It also details the gradient buildup, saving processes, and evaluation approach and steps. The PEGASUS model, training arguments, tokenizer, data collator, and datasets for validation and training are built along with a Trainer instance. The process of fine-tuning is initiated by calling the train() method. ROUGE measures are used to assess the model's performance on the test dataset following training. These metrics are calculated via the test-ds function, and the output is saved in a Pandas DataFrame. Both the tokenizer and the refined model are stored for further use. To show how to use the model with a test case, the dataset is reloaded. The tokenizer and saved model are used to reinitialize the summarization pipeline. In order to compare the efficiency of the refined model with the test data, the code prints the article, reference summary, and summary created by the model at the end.

**Video Summarization:** Takes a YouTube video, extracts its transcript, and summarizes it. It starts by obtaining the video ID from the URL and then uses YouTubeTranscriptApi to obtain the transcript. After that, the transcript is joined together to form a single string. For processing, the text is split up into 1000 character pieces. Each chunk is summarized using a pipeline from the Hugging Face Transformers library. The summaries are concatenated after being kept in a list. The function then outputs the summary text's length and content.

**Image Summarization:** It first receives an image with OpenCV and then displays it using Matplotlib. After that, text is extracted from the image using Tesseract OCR. The method uses a pre-trained Pegasus model (google/pegasus-cnn-dailymail) to summarize the extracted text after it has received the text. After tokenizing the text, the model creates a brief synopsis by applying certain parameters. The text summary is presented at the end. This method combines NLP and computer vision techniques to automatically extract text from photos and summarize it.

# 5 Implementation

To apply the code on Kaggle using a GPU (T4*2) with the Hugging Face datasets and transformers libraries, first install the necessary packages (transformers, datasets, rouge score, etc.). Use the Huggingface Samsum dataset, and use AutoTokenizer to tokenize the articles and summaries from the google/pegasus-cnn-dailymail model. To make sure the model runs on a CUDA device, transfer it to the GPU. Using pipeline (summarization"), create a summarization pipeline and test it by creating a summary for an example article from the test set. To assess the quality of the generated summaries, use the function to calculate rouge scores. This function uses the model to create summaries of the dataset, processes them in batches, and then compares them to the reference summaries. Establish a trainer with the proper training arguments, such as batch size, learning rate, and assessment approach, if you want to fine-tune the model. Usage model.savepretrained() and tokenizer.save-pretrained() to store the model and tokenizer

for later usage after training.Lastly, create test article summaries using the saved model and evaluate performance by contrasting them with reference summaries. Using this method will guarantee that you can use and assess the Pegasus model for summarizing jobs on Kaggle in an efficient manner.

**Video Summarization:** The code is executed in a Kaggle notebook and extracts the transcript from a YouTube video using the youtube-transcript-api function. The transcript is retrieved using the video ID, which is determined by the video's URL. Next, due to input size limits, the complete transcript text is processed in batches of about 1000 characters. Each text chunk is summarized succinctly using the pipeline('summarization') function from the Hugging Face transformers library. A thorough synopsis of the full video transcript is produced by combining the summaries that have been produced. To accelerate the summarization procedure, Kaggle GPU acceleration is used.

**Image Summarization:** To use image summarization in a Kaggle notebook, first install pytesseract, opencv-python, and the Tesseract executable. Configure Pytesseract to use the Tesseract executable by importing libraries such as pytesseract, cv2, matplotlib, and transformers. Upload an image to a dataset, use matplotlib to display it, and then use cv2.imread() to read it. Use the OCR method to extract text from the image. After that, load the Pegasus model and tokenizer from Hugging Face, tokenize the text that has been extracted, and use the model to create a summary. Print the text summary at the end.

# 6 Evaluation

```
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

Reference Summary:
Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry.

Model Summary:
Amanda can't find Betty's number. Larry called Betty last time they were at the park together. Hannah wants Amanda to text Larry. Amanda will text Larry.
```

Figure 5: Dataset Summary

This project's goal was to enhance the Pegasus model's summarization performance by fine-tuning it with the SAMSum dataset. When the model's ROUGE-1 score was 0.015568, it meant that the summary quality was low. Following adjustments, the

| | rouge1 | rouge2 | rougeL | rougeLsum |
|---|---|---|---|---|
| **pegasus** | 0.015568 | 0.000294 | 0.015536 | 0.015566 |

Figure 6: Before training rouge

ROUGE-1 score increased to 0.018595, indicating that the model's capacity to produce accurate and pertinent summaries had improved. This improvement shows that the model's summarization abilities were successfully improved by fine-tuning, producing outputs that were more accurate and logical. As a result, the summaries that are produced now more closely match the source texts, demonstrating the model's improved performance.

| | rouge1 | rouge2 | rougeL | rougeLsum |
|---|---|---|---|---|
| pegasus | 0.018595 | 0.000294 | 0.018437 | 0.018469 |

Figure 7: After training rouge

## 6.1 Experiment / Case Study 1

The transcript of the video is 3,240 words long. Creating a succinct synopsis by summarizing it. This method preserves important details from the video content while guaranteeing the summarizer receives manageable inputs.


Figure 8: Youtube video

## 6.2 Experiment / Case Study 2

Using Pytesseract, first extract the text from the image. Next, import the Hugging Face tokenizer and Pegasus model. Tokenize the text and use the model to provide a summary. To obtain a succinct summary of the original image content, print the summarized text at the end.

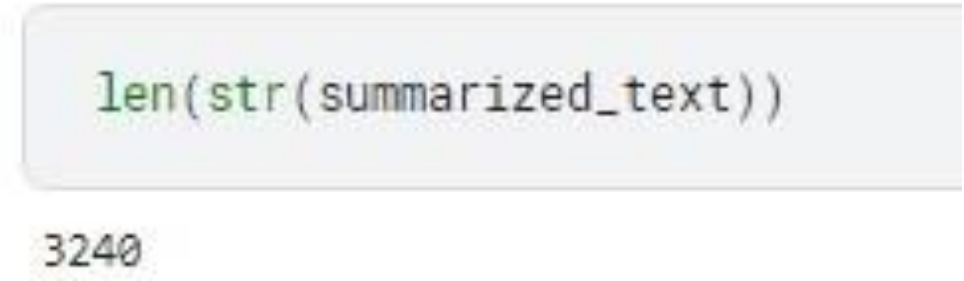


Figure 9: Length of video

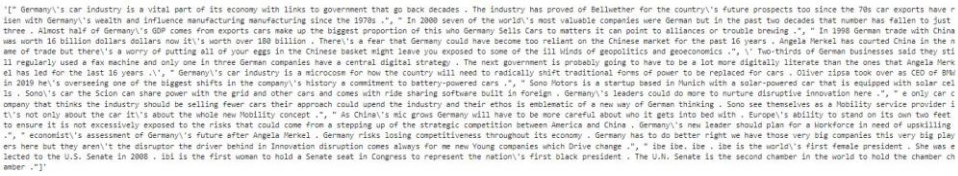


Figure 10: Youtube video summary

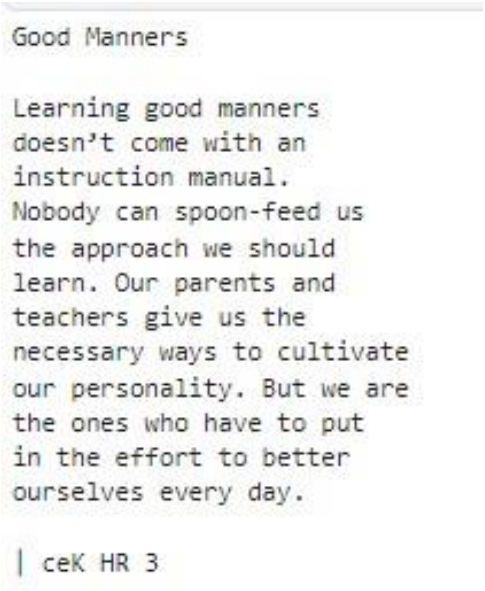


Figure 11: Extracted text from image


Figure 12: Image summary

# 7 Conclusion and Future Work

The goals were to assess how well the SAMSum dataset can be used to fine-tune the Pegasus model and to examine the ways in which the features of the dataset impact the quality of the summarization. Pegasus's summarization quality was shown to increase with fine-tuning; the model produced dialogue summaries that were more cohesive and pertinent to the context. The SAMSum dataset's size and diversity were important factors since larger and more diverse datasets performed better and allowed for better generalization. mention the substantial increase in summary quality attained through fine-tuning and the significance of dataset diversity in boosting model generalization. Nonetheless, managing intricate conversation structures and upholding the coherence of the summary can present difficulties. Future work should include the creation of a user interface for video and picture summarization, which would broaden the scope of the current study to include more modalities. This can entail investigating multiple models and datasets to evaluate their efficacy in relation to diverse content kinds. Furthermore, this method could be commercialized by being integrated into media content management systems or customer service platforms, where precise and effective summarizing is crucial.

To conclude, the project effectively optimized the Pegasus model for discussion summarizing, emphasizing the significance of dataset characteristics and creating opportunities for more study and business development.

# References


Asmitha, M., Danda, A., Bysani, H., Singh, R. P. and Kanchan, S. (2024). Automation of text summarization using hugging face nlp, *2024 5th International Conference for Emerging Technology (INCET)*, IEEE, pp. 1–7.

Etemad, A. G., Abidi, A. I. and Chhabra, M. (2021). Fine-tuned t5 for abstractive summarization, *International Journal of Performability Engineering* **17**(10): 900.

Mahalakshmi, P. and Fatima, N. S. (2022). Summarization of text and image captioning in information retrieval using deep learning techniques, *IEEE Access* **10**: 18289–18297.

Mittal, C., Yadav, K. D. and Kumar, V. (2023). Exploring the effectiveness of different abstractive text summarization models–a comparative study, *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, IEEE, pp. 209–214.

Panthagani, V. B., Duggempudi, V. D. R., Kommera, N. L. and Vakkalagadda, N. (2024). Youtube transcript summarizer, *2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, IEEE, pp. 891–898.

Rydning, D. R.-J. G.-J., Reinsel, J. and Gantz, J. (2018). The digitization of the world from edge to core, *Framingham: International Data Corporation* **16**: 1–28.

Shakil, H., Farooq, A. and Kalita, J. (2024). Abstractive text summarization: State of the art, challenges, and improvements, *Neurocomputing* p. 128255.

Srivastava, A., Hazela, B., Singh, S. and Singh, V. (2024). Streamlining information: Creating youtube video summarizer using machine learning, *Journal of Management and Service Science (JMSS)* pp. 1–9.

Sun, W., Fang, C., Chen, Y., Zhang, Q., Tao, G., You, Y., Han, T., Ge, Y., Hu, Y., Luo, B. et al. (2024). An extractive-and-abstractive framework for source code summarization, *ACM Transactions on Software Engineering and Methodology* **33**(3): 1–39.

Zhang, J., Zhao, Y., Saleh, M. and Liu, P. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, *International conference on machine learning*, PMLR, pp. 11328–11339.