FINAL PROJECT DEEP LEARNING

SUMA: AI News and Video Summarizer
using Fine-Tuned BART Model

Oleh:

2702355302	Kevin Joseph Handoyo
2702331990	I Kadek Defa Danuarta
2702278625	Kristian Binsar Pardamean Pasaribu

5th Semester
Deep Learning - COMP6826001 - LC01

Table of Contents

ABSTRACT

This study introduces SUMA AI, a multimodal system developed to address the growing challenge of information overload in digital media by providing automatic summarization for both news articles and video content. The system combines Whisper for speech transcription with a fine-tuned BART model for abstractive text generation, creating an integrated pipeline capable of transforming long and complex inputs into concise and coherent summaries. The BART model is fine tuned using the CNN and DailyMail dataset and demonstrates strong performance improvements across ROUGE metrics when compared with its pretrained baseline. System implementation employs FastAPI for backend processing and Next.js for the user interface, enabling efficient handling of text inputs, audio extraction, speech transcription, and summary generation. Experimental results show that SUMA AI produces summaries that preserve essential facts and maintain structural consistency for both textual and spoken content. The system offers a practical and accessible tool for rapid information consumption and provides a foundation for future work in multilingual summarization, extended multimodal understanding, and more scalable deployment infrastructures.

INTRODUCTION

Today, the rapid growth of digital content has made it increasingly difficult for users to stay informed without feeling overwhelmed. News outlets, social media platforms, and video-based services produce continuous streams of text, audio, and visuals, contributing to widespread information overload [1]. This constant flow often leads to cognitive fatigue and a growing preference for tools that provide quick access to essential points [2]. Automated summarization has therefore become an important solution for helping users extract meaningful insights without the effort of reading or watching full-length material [3].

Despite these advancements, accessing relevant information efficiently remains a challenge. Manual skimming frequently leads to incomplete understanding, especially when users face time constraints. The increasing complexity of digital content also requires technologies capable of handling long texts and multimedia inputs. Existing systems tend to focus on a single modality, either text or audio, while modern information consumption often involves combined sources such as news videos and podcast clips [4]. These limitations highlight the need for multimodal approaches that can deliver accurate and concise summaries from heterogeneous data.

The goal of SUMA AI is to address this gap by integrating natural language processing and speech-to-text methods into a unified summarization platform. Through the combination of Whisper for accurate transcription and a fine-tuned BART model for abstractive summarization, SUMA AI is designed to convert both news articles and video content into coherent, condensed summaries [5]. This approach is intended to help users access essential information quickly and effectively.

The significance of this system lies in its ability to reduce cognitive overload, increase comprehension efficiency, and improve the accessibility of information across modalities. By merging state-of-the-art NLP and ASR technologies, SUMA AI demonstrates how multimodal summarization can offer practical and reliable support for everyday information needs. In doing so, it contributes to a broader effort to manage information overload in modern digital environments.

LITERATURE REVIEW

Text Summarization

Text summarization is the process of condensing lengthy text into shorter, coherent versions that retain essential meaning and context. It is broadly classified into two approaches: extractive and abstractive. While extractive methods simply select key sentences from the source, abstractive methods create new sentences that often naturally capture the source message better [6]. Summarization research has been taken a step forward with transformer-based architectures, especially models like BERTSum, T5, and BART, which learn semantic representations through large-scale pretraining with unprecedented performance [7]. The CNN/DailyMail dataset has been widely adopted for summarization system evaluations, thus serving as a standard testbed for assessing fluency and factual consistency in automatically generated summaries [8]. Abstractive summarization, especially by BART, works very well for news summarization, as it not only captures contextual relationships between entities but also generates coherent paraphrased snippets free of redundancy—two salient features very essential for a time-critical task of news summarization [9].

Video Summarization and Speech-to-Text

Advances in Automatic Speech Recognition have enabled spoken language to be accurately transcribed into text, providing a solid backbone on which most video summarization tasks rely. State-of-the-art ASR systems, including OpenAI's Whisper, have shown strong robustness against diverse accents, background noise, and long recordings [10]. Whisper uses a multilingual encoder–decoder architecture that empowers high-fidelity transcription with very low error rates [11]. Domain variability, overlapping speech, and acoustic distortion remain challenges for audio-based summarization [12]. Integrating ASR with natural language processing models like BART has enabled end-to-end automatic transcription, content analysis, and summarization of spoken content. Examples of such are found in multimodal frameworks like WhisperSum and similar research works [13]. Integrating both ASR and text summarization will, for the first time, bridge the wide gap between speech understanding and textual summarization, potentially leading to more efficient analysis of multimedia content [11].

Transformer-Based Models for Summarization

Transformer-based architectures have revolutionized natural language understanding and generation by incorporating attention mechanisms that can model long-range dependencies between words [7]. The encoder–decoder framework in BART, among other variants, enables contextual encoding and semantic-rich summary decoding [9]. BART combines the best of both worlds: bidirectional encoding (as in BERT) and autoregressive decoding (as in GPT), thus making it very effective for abstractive summarization tasks [6]. Comparative studies have presented BART to outperform models like T5 and Pegasus on news summarization benchmarks owing to its denoising pretraining and adaptability to domain-specific data [8]. In particular, it is the capacity of BART to deal with diverse linguistic structures while maintaining factual coherence that best equips this model to generate concise yet informative summaries from complex news datasets [9].

Supporting Open-Source Tools and Frameworks

Modern summarization systems are typically based on open-source frameworks, which allow for ease of deployment and scalability. FastAPI provides a lightweight, high-performance backend to deploy summarization models with real-time serving capabilities [12]. On the frontend, Next.js provides a responsive, interactive user interface with dynamic content rendering suitable for AI-driven applications [13]. In addition, the HuggingFace Transformers library is one of the most popular deep learning libraries for NLP research, providing a variety of pre-trained models like BART and Whisper, whose fine-tuning and seamless deployment within summarization workflows are straightforward to be adopted [11]. These open-source tools collectively provide flexible, modular, and cost-efficient development of intelligent summarization platforms according to the architecture shown in systems like SUMA AI [10].

Existing Related Systems

Most of these, including currently working systems like TL;DR generators, online news summarizers, and auto-captioning tools, try to solve the problem of information overload but still have a very limited scope for their application [7]. Most applications use only text-based summarization and do not support video or audio content, hence limiting their application in a complete multimedia environment [6]. In addition, most such tools generate very generic output with low contextual accuracy or miss the important semantic information [9]. ASR and NLP-based integrated systems are still at the nascent stage of development, and the integrated understanding of multimodal input is rarely realized [10]. SUMA AI overcomes these deficits by integrating Whisper's speech recognition with a fine-tuned BART model for abstractive summarization, proposing a united multimodal framework that can summarize written and spoken content with higher precision and accessibility [11].

METHODOLOGY

Dataset

Dataset Source

This study uses the CNN/DailyMail dataset, a widely adopted benchmark for abstractive news summarization. The dataset contains pairs of full news articles and their corresponding highlights, which serve as ground-truth summaries. In this project, the dataset is accessed through the HuggingFace Datasets library in Parquet format, organized into train, validation, and test directories.

CNN/DailyMail is selected because of its extensive use in summarization research, its well-structured article–summary pairs, and its suitability for fine-tuning transformer-based models such as BART.

Dataset Size

According to the dataset loading steps in the notebook, the corpus is divided into:

- Training set: a sampled subset of 95705 articles
- Validation set: a sampled subset of 13,368 articles
- Test set: a sampled subset of 11,490 articles

Each partition is converted into a pandas DataFrame for exploratory analysis. Basic structural checks confirm the presence of the required fields and show no missing values in the main textual columns.

Data Characteristics

The Parquet files contain structured fields relevant to summarization tasks, primarily:

- article: the full text of the news article
- highlights: the reference summary

The notebook performs descriptive analysis on text length, revealing typical CNN/DailyMail characteristics: articles usually span hundreds of words, while summaries are significantly shorter, often limited to a few sentences. Length distributions are visualized through histograms, and the relationship between article length and summary length is explored using statistical correlation.

Additional linguistic analysis includes examination of frequent words, bigrams, and trigrams extracted from a sample of the training corpus, offering insights into stylistic and topical patterns present in the dataset. Word clouds are also generated to visualize dominant terms in both article bodies and summaries.

Rationale for Dataset Selection

Several factors motivate the use of the CNN/DailyMail dataset:

- Benchmark Standard: It is one of the most widely used datasets in abstractive summarization research, enabling meaningful comparison with prior work.
- Clear Input–Output Structure: The article–summary format is consistent and well suited for training sequence-to-sequence models.

- Large Corpus Size: Its scale provides sufficient variability for fine-tuning large transformer models.
- Relevance to News Summarization: The dataset aligns closely with the objectives of SUMA AI, which focuses on generating concise summaries from long news content.

These properties make CNN/DailyMail an appropriate choice for developing and evaluating the summarization components of SUMA AI.

Preprocessing

Loading and Normalization

The preprocessing workflow begins by loading the CNN/DailyMail dataset in Parquet format through the HuggingFace load_dataset interface. The dataset consists of three splits: train, validation, and test, where the validation and test sets are downsampled to 2,000 and 1,500 samples respectively for computational efficiency. After loading, each split is treated as a HuggingFace Dataset object, which enables efficient mapping operations and batch preprocessing.

As part of normalization, the dataset retains only the two relevant textual fields: article and highlights. No missing values are present in these fields, so no imputation is required. The dataset is then prepared for token-level processing by ensuring that its columns align with the input format expected by the BART tokenizer and model.

Tokenization

Tokenization is performed using the BART-base tokenizer. Each news article is converted into a sequence of input tokens that the model can process. The tokenizer applies standard text normalization, such as lowercasing, subword segmentation (Byte-Pair Encoding), and insertion of special tokens, to transform raw text into numerical token IDs.

For the target summaries, the tokenizer is temporarily switched into "target mode," enabling the model to treat summary tokens as decoder inputs. This separation ensures that BART's encoder processes the article while the decoder learns to generate the corresponding summary.

Truncation and Padding

To maintain batch consistency and avoid excessively long sequences, both article inputs and reference summaries are subject to truncation and padding:

- Maximum input length: 768 tokens
- Maximum summary length: 256 tokens

Articles exceeding the maximum length are truncated from the end, while shorter articles are padded. The same approach is applied to summaries. Padding tokens in the summary labels are replaced with the value –100, which instructs the loss function to ignore padded positions during training. This prevents the model from being penalized for predicting tokens in locations that correspond to padding rather than meaningful content.

Filtering

During preprocessing, the dataset implicitly filters out unsuitable samples by truncating overly long articles and summaries. While the code does not explicitly remove samples based on length thresholds, this truncation process effectively prevents extremely long documents from dominating training or exceeding memory capacity. This contributes to stable training behavior and ensures that the model receives consistent, manageable input sizes.

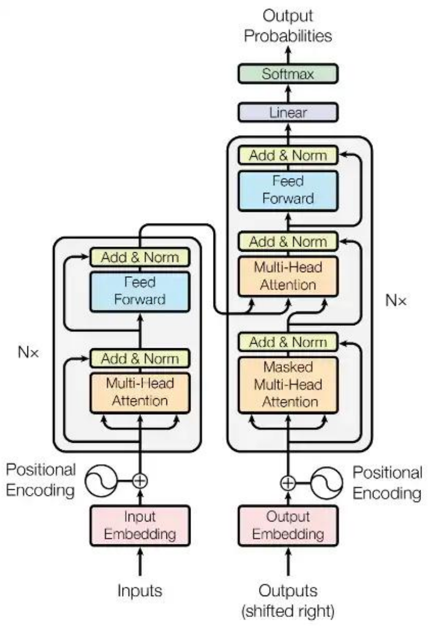
Final Tokenized Dataset Construction

Once tokenization, truncation, padding, and label masking are completed, each dataset split is transformed using a mapping function. The mapping removes the original text columns and retains only the model-ready features: token IDs, attention masks, and masked labels.

The resulting tokenized datasets are then passed to the Seq2SeqTrainer, enabling efficient batching, GPU acceleration through FP16 precision, and integration with evaluation metrics such as ROUGE score.

Model Architecture

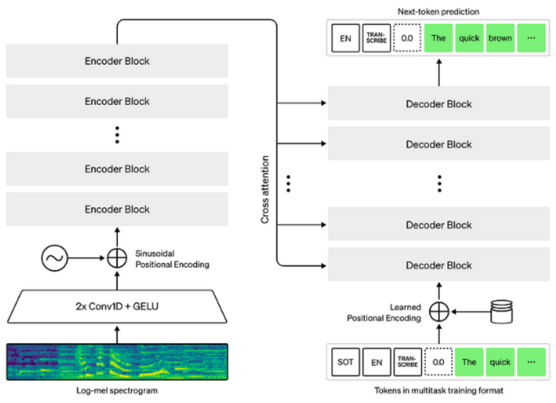
BART for Abstractive Summarization



The core text summarization component of SUMA AI is built on BART (Bidirectional and Auto-Regressive Transformers), a sequence-to-sequence encoder–decoder architecture. The encoder processes the full input article, capturing long-range dependencies through bidirectional attention, while the decoder generates the summary autoregressively, producing one token at a time based on previously generated output.

The model uses pretrained BART-base weights, which have been trained on large-scale corpora to learn general linguistic representations. SUMA AI fine-tunes these pretrained weights specifically on the CNN/DailyMail dataset, enabling the model to learn the structure, compression patterns, and stylistic characteristics of news summaries. This fine-tuning process allows BART to produce abstractive summaries, meaning the model generates new phrasing rather than simply extracting sentences from the source text. The result is a concise, coherent summary that captures the key information from the original article.

Whisper for Speech-to-Text



To support video summarization, SUMA AI employs Whisper, an automatic speech recognition (ASR) model developed by OpenAI. The system uses the small variant of Whisper, which offers an optimal balance between accuracy and computational efficiency, making it suitable for CPU-based inference.

Whisper operates by converting an audio waveform into a sequence of text tokens. It processes the input through a transformer encoder that captures acoustic patterns and contextual relationships, followed by a decoder that produces the transcription. Whisper is chosen due to its robust multilingual performance, ability to handle noisy or imperfect audio, and consistent stability across diverse recording conditions. These characteristics make it a reliable component for extracting textual information from news videos, interviews, and spoken content.

System Architecture Overview

SUMA AI integrates both components (Whisper and Fine-Tuned BART) into a unified multimodal summarization pipeline. The system accepts two types of user inputs: a news article URL or a video file (e.g., MP4).

- When processing video content, the system extracts the audio track and forwards it to Whisper. Whisper transcribes the speech into text, creating an intermediate textual representation of the video's spoken content.
- When processing textual news articles, the raw article text is passed directly to the summarization module.

In both cases, the textual content, whether extracted by Whisper or provided directly by the user, is then fed into the BART summarizer. BART generates an abstractive summary that distills the essential information into a concise, readable format. Finally, the system returns the summary through the user interface, providing an efficient, multimodal understanding of the source material.

This integrated architecture enables SUMA AI to process both written and audiovisual content, offering flexible and robust summarization capabilities for modern information environments.

Training Setup

Environment

BART model is fine-tuned using PyTorch and the HuggingFace Transformers library. Training is executed on an NVIDIA RTX 3060 GPU, which supports mixed-precision FP16 for faster computation and reduced memory usage.

Training Parameters

The training configuration includes:

- Batch size: 2 per device, with gradient accumulation of 4 steps (equivalent to batch size 8).
- Learning rate: 3e-5 using a linear schedule.
- Epochs: 6 total fine-tuning epochs.
- Warmup steps: handled implicitly by HuggingFace's scheduler.
- Truncation lengths: max input 768 tokens, max summary 256 tokens.
- Mixed precision: FP16 enabled for speed and memory efficiency.
- Gradient checkpointing: enabled to reduce VRAM usage.

Loss Function

The model is optimized using cross-entropy loss, where padded positions in the target sequence are masked with -100 so they do not contribute to the loss.

Optimizer

Training uses Adafactor, commonly applied in transformer fine-tuning due to its memory-efficient updates and stable training without needing momentum states.

Validation Strategy

The model is evaluated on the validation set at the end of each epoch, using ROUGE metrics. Early stopping is enabled to prevent overfitting if validation performance stagnates.

Checkpointing and Model Hosting

The best checkpoint is saved and uploaded to HuggingFace Hub for hosting.

Evaluation Metrics

Evaluation uses four common summarization metrics:

- ROUGE-1: measures unigram (single-word) overlap between prediction and reference. Reflects how much basic content is retained.
- ROUGE-2: measures bigram overlap. Captures fluency and short-phrase accuracy.
- ROUGE-L: measures the longest common subsequence (LCS). Evaluates overall structural similarity and coherence.
- ROUGE-Lsum: a summary-level variant of ROUGE-L that computes LCS across entire summaries rather than sentence-by-sentence.

These scores provide a quantitative assessment of how well the generated summary matches the reference highlights.

IMPLEMENTATION & RESULT

System Implementation Details

Overall System Workflow

SUMA AI integrates both text summarization and video-based summarization in a unified multimodal pipeline. The workflow consists of:

1. Input
   - User provides a news article URL or uploads a video file (MP4/MKV/MOV/WAV).
2. Preprocessing
   - For URLs: article text is scraped and cleaned.
   - For videos: audio is extracted and normalized.
3. Speech-to-Text (Whisper Small)
   - Whisper transcribes the audio into text with high robustness to noise.
4. Abstractive Summarization (Fine-tuned BART)
   - The transcribed text or article content is passed to the BART summarizer.
   - Output is an abstractive summary representing the essential information.
5. Output Delivery
   - The summary is returned to the frontend and displayed to the user.

This pipeline supports multimodal input, textual articles and audiovisual sources, while producing a unified textual summary.

Backend Implementation (FastAPI)

The backend exposes two main endpoints:

- /summarize-news: accepts news link
- /summarize-video: accepts MP4/MKV/MOV/WAV uploads

Backend processes include:

- Static loading of Whisper small and fine-tuned BART to avoid repeated initialization.
- Handling uploaded files and saving them temporarily.
- Running audio extraction and Whisper transcription.
- Running BART inference to generate summaries.
- Returning results in JSON format.

Frontend Implementation (Next.js)

The frontend provides an interactive user interface with:

- Video upload and URL input components
- Progress bars and shimmer loading skeletons to indicate process stages
- Display of summaries with clean formatting
- Communication with the backend uses HTTPS POST requests.

The user experience is optimized through real-time updates, such as:

- "Generating summary…"

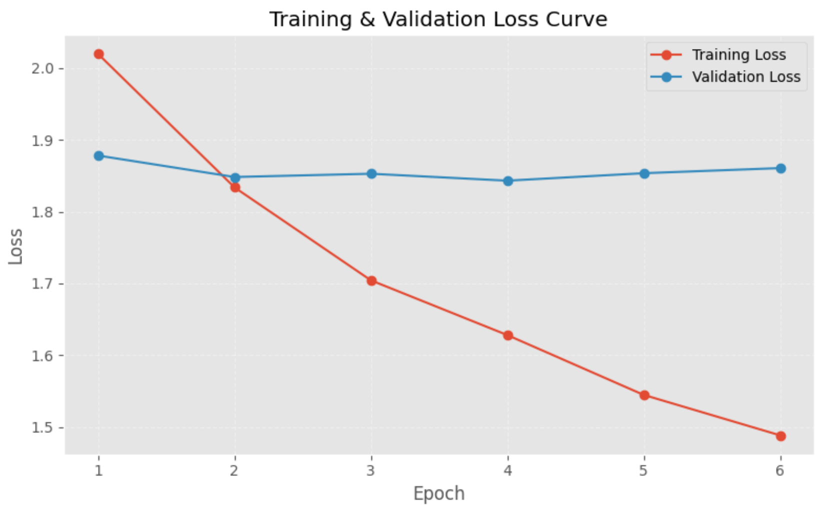
Deployment Setup
The system is deployed on cloud platforms:

- Frontend: Vercel (Next.js)
- Backend: HuggingFace Spaces (FastAPI)
- Model Hosting: HuggingFace Hub (BART fine-tuned model, Whisper)

This architecture enables scalable inference and ensures models are easily accessible for production.

Experimental Results

Training Curve



During fine-tuning of BART:

- Training loss decreases steadily across epochs, indicating consistent learning.
- Validation loss stabilizes earlier and plateaus, showing no severe overfitting due to early stopping and weight decay.
- The model remains stable across the 6-epoch training schedule.

The curves demonstrate that the model benefits from domain-specific fine-tuning while maintaining generalization.

ROUGE Evaluation
Performance on the held-out test set is summarized below:

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
BART-NewsSummarizer (Fine-Tuned)	35,9819	15,5956	26,2151	33,2041
BART (Baseline)	30,5177	12,7930	19,9842	24,5837

Observations:

- Fine-tuned BART-NewsSummarizer outperforms the pretrained model across all metrics.
- ROUGE-1 and ROUGE-Lsum show the most improvement, indicating stronger phrase-level consistency and overall coherence.

Qualitative Summaries
Example:

- Original article: Long-form news article (200–800 words).
- Reference summary (highlights): 2–4 sentence human-written summary.
- Model summary: Concise abstract capturing main entities, events, and outcomes.

Analysis:

- The model correctly identifies central themes and events.
- Occasionally introduces minor paraphrasing but maintains factual consistency.
- Low hallucination tendency due to supervised fine-tuning on structured news data.

End-to-End Video Summarization Example

Case Study:

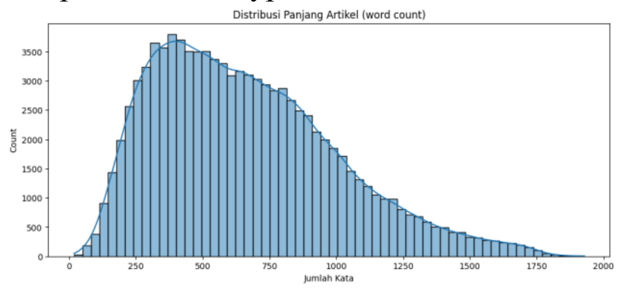
- Input:
  - Link: https://edition.cnn.com/2025/12/10/politics/trump-gold-card-1-million-dollar-visa
  - Title: 'Trump Gold Card' launches, offering expedited immigration pathway with a $1 million fee
- BART-NewsSummarizer Summary:
  - President Donald Trump officially launches his "gold card" visa. The visa will allow foreigners to pay $1 million to expedite their visa application. Applicants will also be able to spend up to 270 days in the United States without U.S. taxes. Trump has suggested millions of gold cards could be sold, and said in February the plan could raise $1 trillion to pay down the national debt. Immigration law experts previously said ending the EB-5 program or significantly changing it would require Congress to act. Despite persistent questions about how the plan will be implemented.

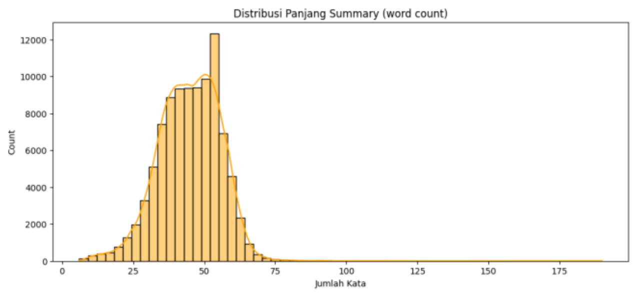
Output summary aligns well with the news' content, demonstrating effective model performance.

Visualizations

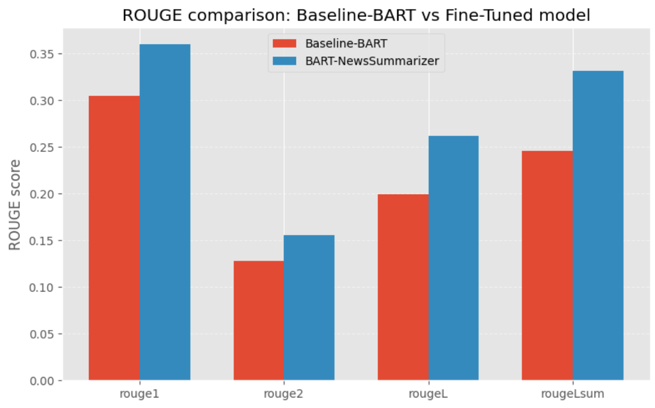
Our notebook provides several key visualizations that support the experimental analysis:

1. Histogram of article vs summary length
   - Shows strong compression ratio typical of news summarization datasets.





2. Evaluation score chart
   - Visual comparison of ROUGE metrics across epochs or model variants.



These visualizations strengthen the justification for model performance and training stability.

DISCUSSION & LIMITATIONS

Discussion

Interpretation of Model Performance

The ROUGE improvements indicate that the fine-tuned BART model successfully captures key information from news articles and produces coherent summaries that align with human-written highlights. This strong performance reflects the structured and factual nature of news text, which fits well with BART's pretrained knowledge.

In real-world usage, the pipeline generalizes reasonably well, though articles with unusual structure or informal writing show minor drops in coherence. For videos, the Whisper to BART pipeline works effectively: Whisper provides stable transcripts, and BART condenses them into readable summaries.

Observed Model Behaviors

During evaluation, BART consistently identifies main events, entities, and causal relationships. It also paraphrases naturally, though sometimes too freely, reducing factual specificity. Whisper performs reliably on English speech, maintaining clarity even with moderate background noise, which helps downstream summarization remain accurate.

Pipeline Efficiency

The end-to-end latency depends on input type: text summarization is fast, while video summarization is bottlenecked by Whisper's transcription time, especially on CPU. The models require moderate GPU resources but remain usable on CPU with increased latency. Audio quality strongly influences final output; noisy or low-bitrate recordings may degrade both transcription and summary accuracy.

Limitations

Model-Related Limitations

BART occasionally produces hallucinated details, especially when inputs are long or ambiguous. The system currently handles only English effectively, limiting real-world applicability. Summaries may become overly compressed, sacrificing nuance.

Whisper Limitations

Whisper struggles with heavy background noise, overlapping speakers, or strong accents. On CPU, transcription is slow, affecting the user experience for long videos.

System Limitations

The system cannot efficiently process extremely long documents due to token length constraints. There is no caching or batching, which increases latency for repeated inputs. The pipeline captures only audio and text, lacking visual understanding of video frames.

Evaluation Limitations

ROUGE metrics assess lexical overlap but do not measure coherence, factual accuracy, or readability. Human evaluation was not conducted, limiting the depth of performance assessment.

CONCLUSION & FUTURE WORK

Conclusion

SUMA AI successfully integrates Whisper and BART into a multimodal summarization system capable of processing both text and video content. The fine-tuned BART model demonstrates substantial performance gains on the CNN/DailyMail dataset, and the system delivers concise summaries that help users understand long content quickly. The deployment setup (FastAPI backend and Next.js frontendsupports responsive interaction, making the system practical for everyday summarization tasks. Overall, SUMA AI provides an efficient pipeline that reduces cognitive load and streamlines information consumption.

Future Work

Improve Summarization Quality

Enhance model performance through domain-specific fine-tuning, RLHF techniques, and better hallucination control.

Support More Language

Adopt multilingual transformer models and upgrade Whisper to larger variants for improved transcription accuracy.

Add Visual Understanding

Integrate CLIP or Vision Transformers to analyze video frames, enabling true multimodal summarization beyond audio and text.

Enhance User Experience

Implement batch summarization, auto-saving, browser extensions, and more responsive UX interactions.

Improve Deployment

Provide GPU-backed APIs for lower latency, consider model distillation to reduce computational cost, and optimize server resources for production-level scalability.

References

[1] G. Poghosyan, “Addressing information overload through text mining across news and social media streams,” in Proceedings of the ACM International Conference on Information and Knowledge Management, 2019, pp. 1–8. https://doi.org/10.1145/3345645.3351105

[2] D. Vogler and F. Meissner, “Tackling the information overload: Using automated content analysis for crisis communication research,” Wiley Online Library, 2022. https://doi.org/10.1002/9781119678953.ch4

[3] B. Palanisamy, A. Chakrabarti, A. Singh, and V. Hassija, “From information overload to lucidity: A survey on leveraging GPTs for systematic summarization of medical and biomedical artifacts,” IEEE Access, 2024. https://doi.org/10.1109/ACCESS.2024.3521596

[4] A. S. Manek, Vineeta, R. P. Shree, and G. H. Sharath, “An intelligent and fast YouTube video content summarizer,” in Proceedings of the International Conference on Intelligent Systems and Applications. Springer, 2025. https://doi.org/10.1007/978-3-032-05373-2_20

[5] N. B. Raut, A. S. Pranesh, and B. Nagulan, “An Extensive Survey on Audio-to-Text and Text Summarization for Video Content,” IEEE Access, 2023. https://doi.org/10.1109/ICIMIA60377.2023.10426376

[6] T. G. Altundogan and M. Karakose, “Transformer-based Multimodal Summarization and Highlight Abstraction Approach for Texts and Speech Audios,” IEEE, 2024. https://doi.org/10.1109/IT61232.2024.10475775

[7] S. Ganguly, S. Mandal, and N. Das, “WhisperSum: Unified Audio-to-Text Summarization,” IEEE, 2024. https://doi.org/10.1109/IACIS61494.2024.10721926

[8] V. Marklynn, A. Sebastian, and Y. L. Tan, “A Framework for Abstractive Summarization of Conversational Meetings,” IEEE Access, 2024. https://doi.org/10.1109/CCWC60891.2024.10427755

[9] S. Zanwar, S. Srikanth, and A. Ghei, “Automated Notes and Question Generation,” IEEE, 2024. https://doi.org/10.1109/SEAI62072.2024.10674458

[10] A. A. Balushi, A. S. Al-Bemani, S. Al Araimi, and G. Balaji, “AI-Driven Multi-Modal Information Synthesis: Integrating PDF Querying, Speech Summarization, and Cross-Language Text Summarization,” Procedia Computer Science, vol. 240, 2025. https://doi.org/10.1016/j.procs.2025.01.663

[11] R. V. Kulkarni, A. Shilimkar, and S. Bandi, “Transcription, Translation and Summarization to Improve Educational Understanding,” IEEE, 2024. https://doi.org/10.1109/ICTBIG64922.2024.10911052

[12] J. J. Shanthamalar, S. Nithya, and A. N. Kumar, “Efficient Business Meeting Summarization: Leveraging Natural Language Processing for Enhanced Productivity,” Springer, 2024. https://doi.org/10.1007/978-981-97-8865-1_17

[13] M. D. M. Rayhan, “Developing an AI-Driven Network Suite,” Master’s Thesis, Tampere University, 2024. https://trepo.tuni.fi/handle/10024/162769

APPENDIX

Team Contribution Statement

This project was completed through the collaborative effort of three team members, each contributing in distinct and essential roles:

- **I Kadek Defa Danuarta** served as the front end developer and UI and UX designer. He was responsible for creating the user interface in Next.js, designing interaction flows, implementing responsive components, and ensuring that the overall user experience was intuitive and visually consistent throughout the system.
- **Kevin Joseph Handoyo** contributed as the AI engineer and backend developer. His work focused on model fine tuning, preprocessing pipelines, and the integration of Whisper and BART into a unified inference workflow. He also developed the FastAPI backend, implemented inference endpoints, and ensured that the AI components operated efficiently in the deployed environment.
- **Kristian Binsar Pardamean Pasaribu** worked as the researcher, report writer, and DevOps engineer. He conducted literature reviews, structured the methodology and analysis, and prepared the written documentation for the final report. In addition, he managed deployment tasks, coordinated the hosting of models and services, and ensured that the system operated reliably across the chosen infrastructure.

Screenshots

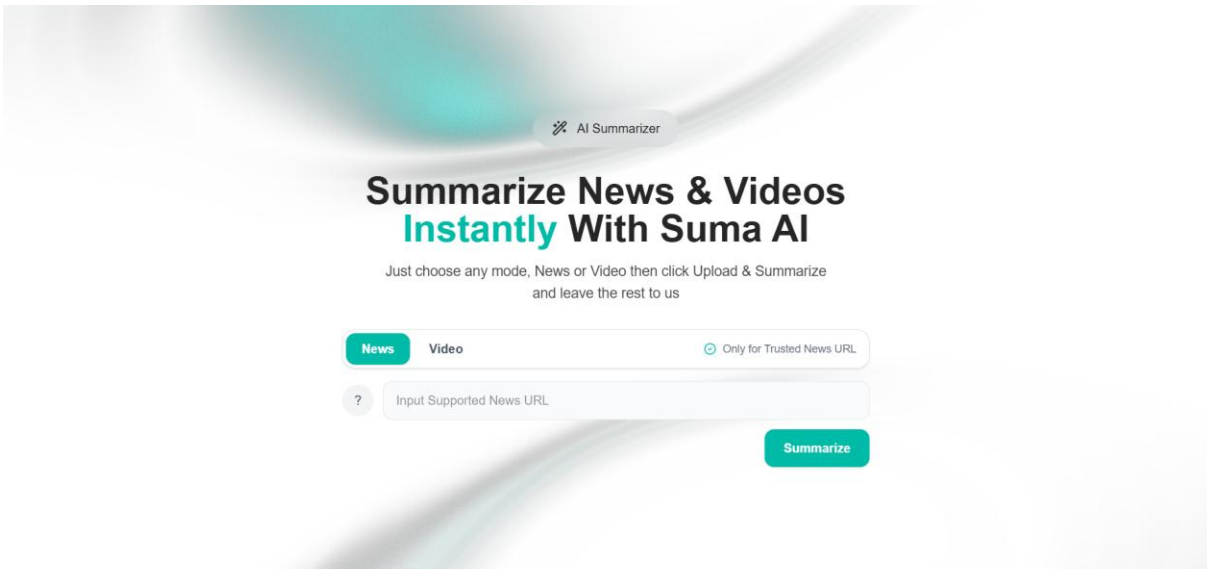

AI Summarizer
Summarize News & Videos
Instantly With Suma AI
Just choose any mode, News or Video then click Upload & Summarize and leave the rest to us
News
Video
Only for Trusted News URL
?
Input Supported News URL
Summarize

Image 1. SUMA AI Landing Page

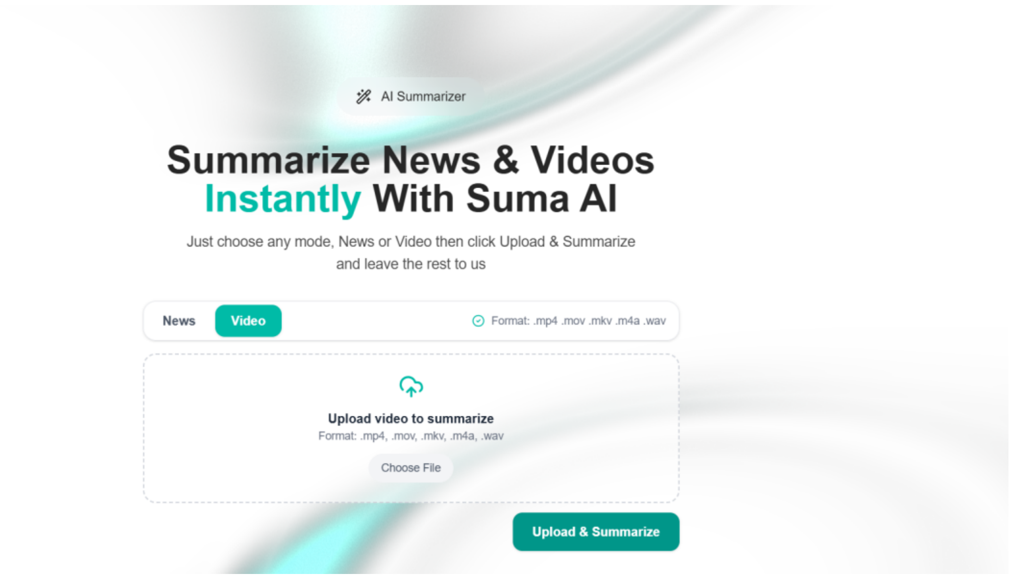

AI Summarizer
Summarize News & Videos
Instantly With Suma AI
Just choose any mode, News or Video then click Upload & Summarize
and leave the rest to us
News
Video
Format: .mp4 .mov .mkv .m4a .wav
Upload video to summarize
Format: .mp4, .mov, .mkv, .m4a, .wav
Choose File
Upload & Summarize

Image 2. Video Upload Section

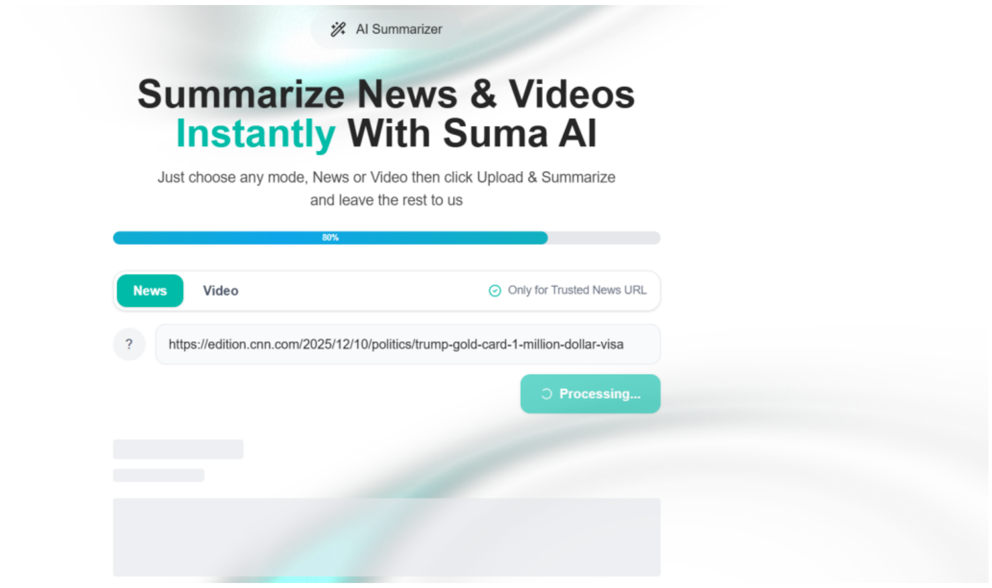
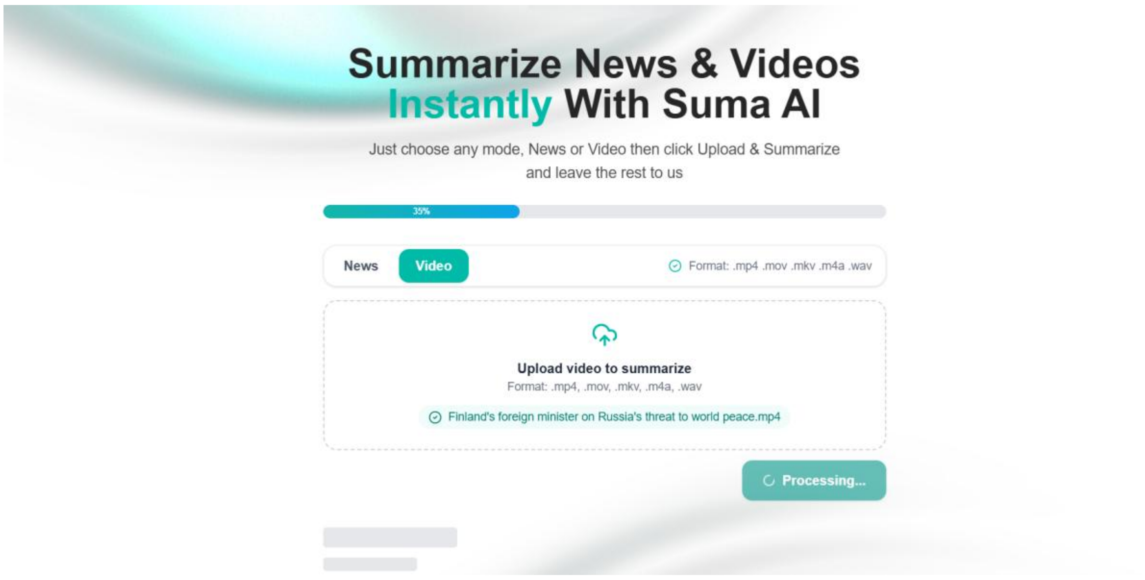
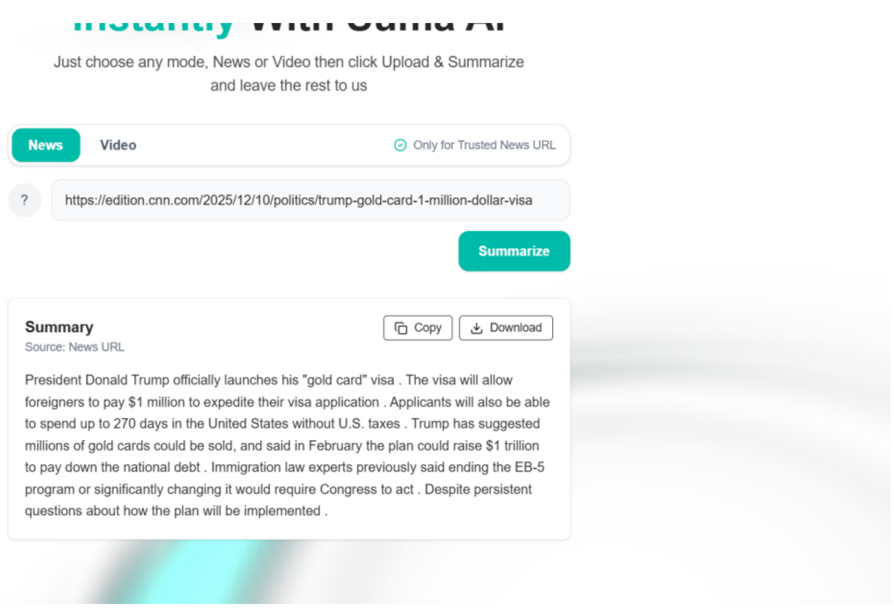

AI Summarizer
Summarize News & Videos
Instantly With Suma AI
Just choose any mode, News or Video then click Upload & Summarize
and leave the rest to us
80%
News
Video
Only for Trusted News URL
?
https://edition.cnn.com/2025/12/10/politics/trump-gold-card-1-million-dollar-visa
Processing...

Image 3. Summarization Loading Screen



Summarize News & Videos Instantly With Suma AI
Just choose any mode, News or Video then click Upload & Summarize and leave the rest to us
35%
News
Video
Format: .mp4 .mov .mkv .m4a .wav
Upload video to summarize
Format: .mp4, .mov, .mkv, .m4a, .wav
Finland's foreign minister on Russia's threat to world peace.mp4
Processing...

Image 4. Video Summarization Loading Screen



Just choose any mode, News or Video then click Upload & Summarize and leave the rest to us
News
Video
Only for Trusted News URL
?
https://edition.cnn.com/2025/12/10/politics/trump-gold-card-1-million-dollar-visa
Summarize
Copy
Download

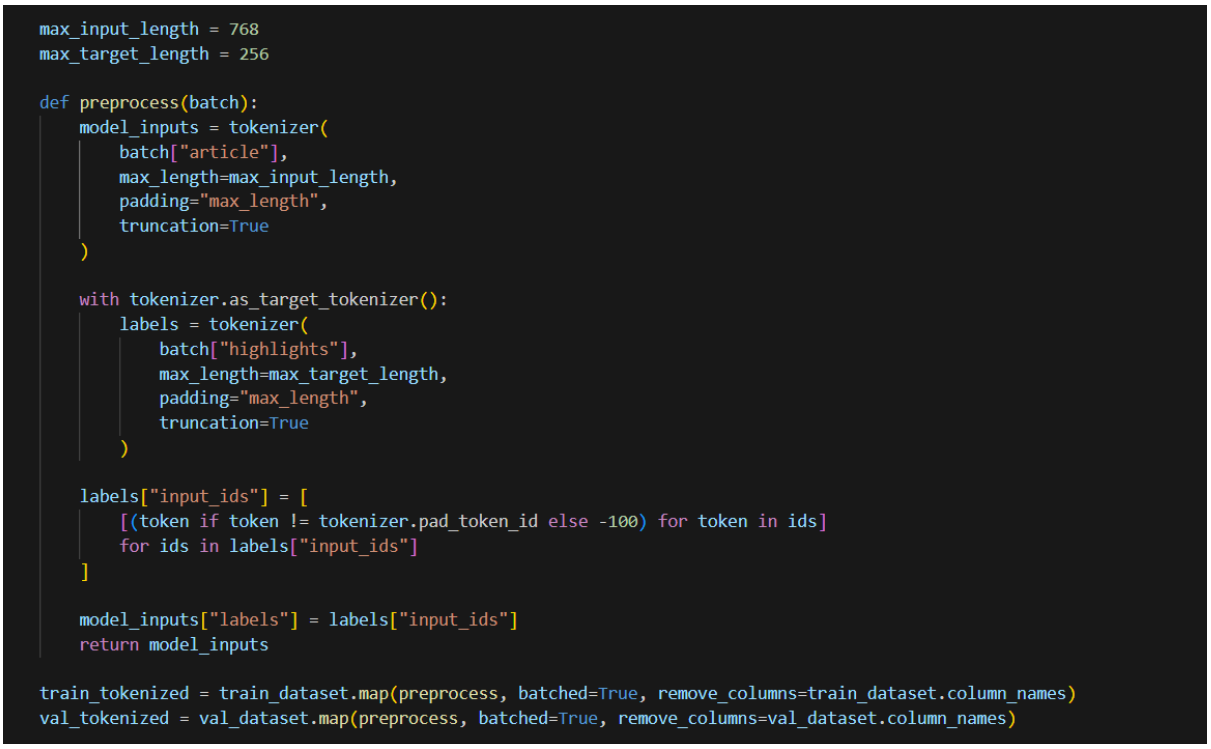
Summary
Source: News URL

President Donald Trump officially launches his "gold card" visa . The visa will allow foreigners to pay $1 million to expedite their visa application . Applicants will also be able to spend up to 270 days in the United States without U.S. taxes . Trump has suggested millions of gold cards could be sold, and said in February the plan could raise $1 trillion to pay down the national debt . Immigration law experts previously said ending the EB-5 program or significantly changing it would require Congress to act . Despite persistent questions about how the plan will be implemented .

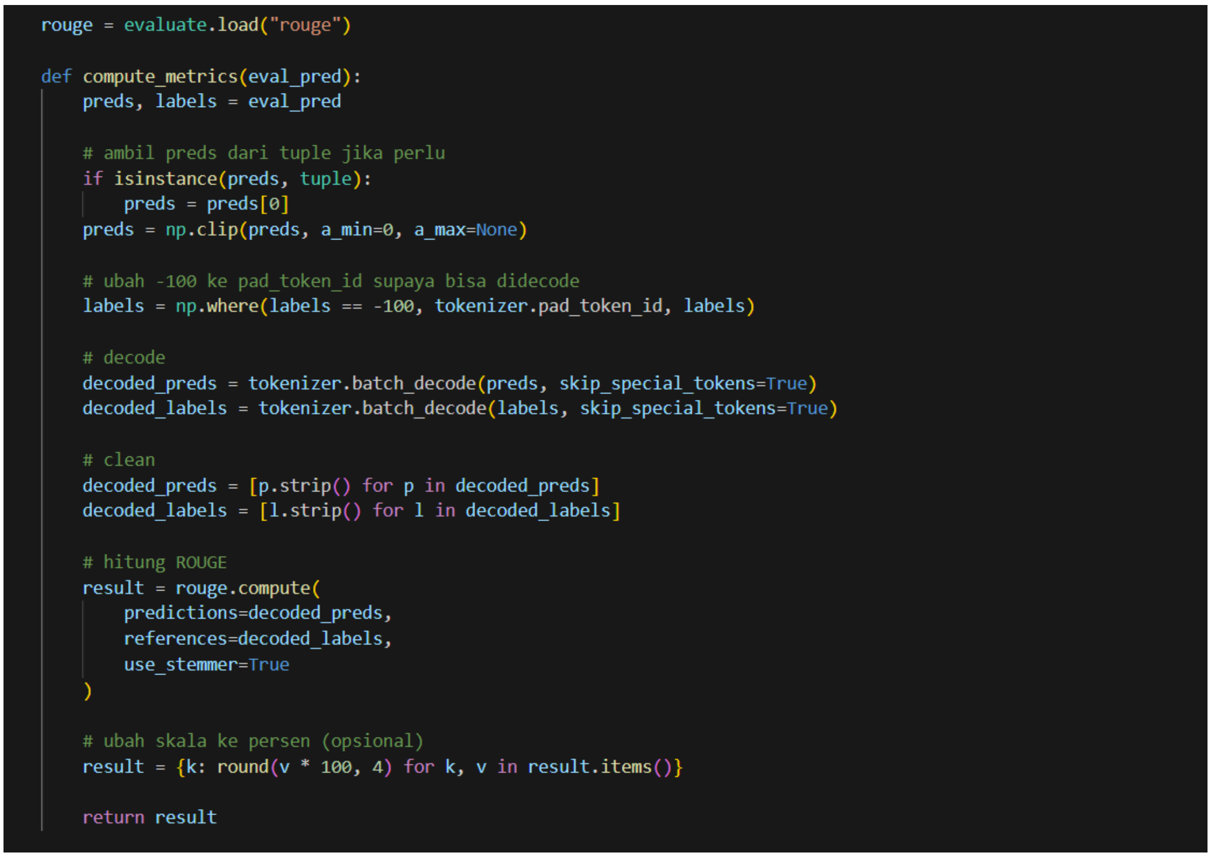
Image 5. Results Visualization.

Code Snippets

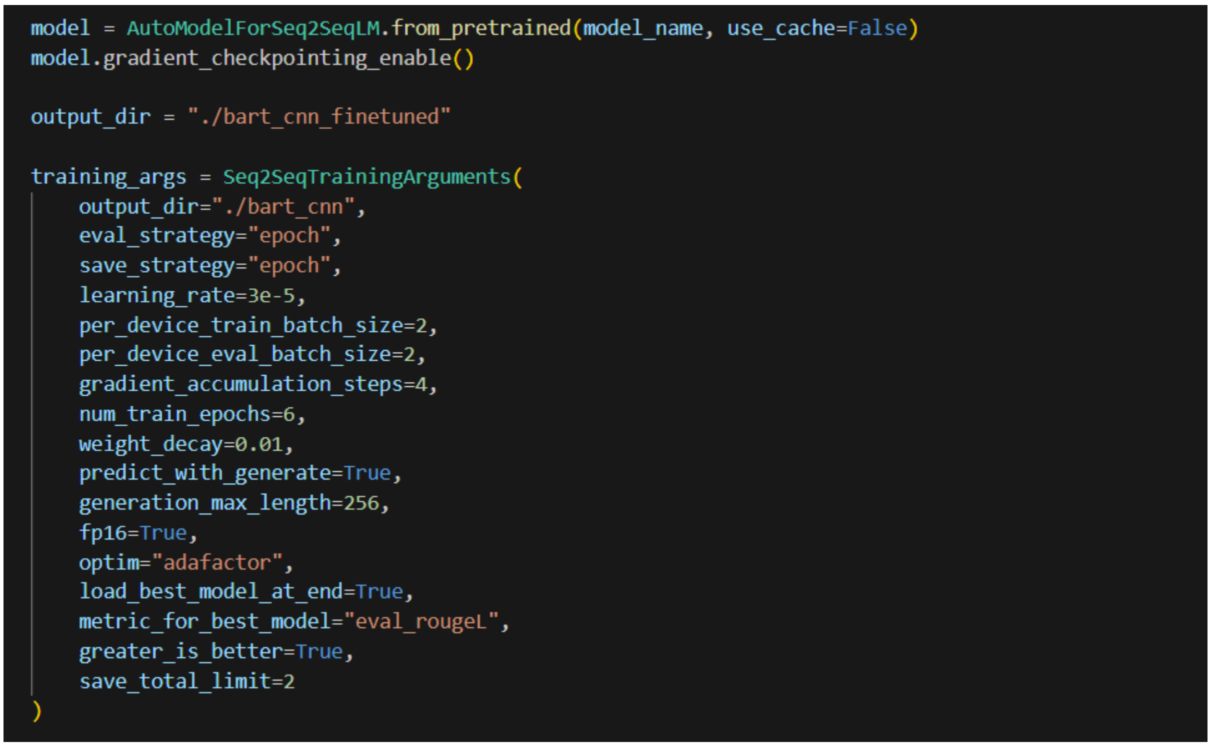
All documentation, source code, and implementation details for this project are available in the public GitHub repository at https://github.com/CicakBelanda/SUMA-AI-Deep-Learning-Final-Project



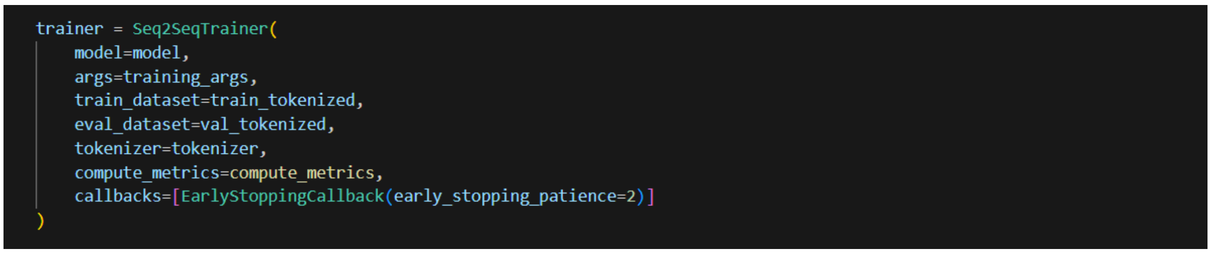
```python
max_input_length = 768
max_target_length = 256

def preprocess(batch):
    model_inputs = tokenizer(
        batch["article"],
        max_length=max_input_length,
        padding="max_length",
        truncation=True
    )

    with tokenizer.as_target_tokenizer():
        labels = tokenizer(
            batch["highlights"],
            max_length=max_target_length,
            padding="max_length",
            truncation=True
        )

    labels["input_ids"] = [
        [(token if token != tokenizer.pad_token_id else -100) for token in ids]
        for ids in labels["input_ids"]
    ]

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

train_tokenized = train_dataset.map(preprocess, batched=True, remove_columns=train_dataset.column_names)
val_tokenized = val_dataset.map(preprocess, batched=True, remove_columns=val_dataset.column_names)
```



```python
rouge = evaluate.load("rouge")

def compute_metrics(eval_pred):
    preds, labels = eval_pred

    # ambil preds dari tuple jika perlu
    if isinstance(preds, tuple):
        preds = preds[0]
    preds = np.clip(preds, a_min=0, a_max=None)

    # ubah -100 ke pad_token_id supaya bisa didecode
    labels = np.where(labels == -100, tokenizer.pad_token_id, labels)

    # decode
    decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    # clean
    decoded_preds = [p.strip() for p in decoded_preds]
    decoded_labels = [l.strip() for l in decoded_labels]

    # hitung ROUGE
    result = rouge.compute(
        predictions=decoded_preds,
        references=decoded_labels,
        use_stemmer=True
    )

    # ubah skala ke persen (opsional)
    result = {k: round(v * 100, 4) for k, v in result.items()}

    return result
```



```
model = AutoModelForSeq2SeqLM.from_pretrained(model_name, use_cache=False)
model.gradient_checkpointing_enable()

output_dir = "./bart_cnn_finetuned"

training_args = Seq2SeqTrainingArguments(
    output_dir="./bart_cnn",
    eval_strategy="epoch",
    save_strategy="epoch",
    learning_rate=3e-5,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=4,
    num_train_epochs=6,
    weight_decay=0.01,
    predict_with_generate=True,
    generation_max_length=256,
    fp16=True,
    optim="adafactor",
    load_best_model_at_end=True,
    metric_for_best_model="eval_rougeL",
    greater_is_better=True,
    save_total_limit=2
)
```



```
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=train_tokenized,
    eval_dataset=val_tokenized,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=2)]
)
```