AI Writer with Text Summarization – Internship Report

Student: Parimal Harish K
Email: harishparimal@gmail.com
Date: December 12, 2025
Domain: AI/ML – Natural Language Processing
GitHub: harishparimal-droid/ai-writer-summarizer

1. Introduction

This internship project focuses on the development of an AI-powered web application for automatic text summarization using the BART (Bidirectional and Auto-Regressive Transformers) model from Hugging Face. The primary objective is to generate concise and readable summaries from long textual content while preserving the original semantic meaning. The project demonstrates a complete NLP pipeline integrated into a simple yet functional full-stack web application.

2. Abstract

The proposed system is a Flask-based web application that accepts long-form text input through an HTML interface and processes it using the facebook/bart-large-cnn transformer model. The application generates abstractive summaries and presents additional evaluation metrics such as original and summarized word counts, compression ratio, and Flesch Reading Ease score. The system was evaluated using short, medium, and long articles to ensure consistent and reliable summarization performance.

3. Tools and Technologies Used

The project was implemented using Python 3.11 and the Flask web framework. The NLP pipeline was built using the Hugging Face Transformers library with PyTorch as the backend and the facebook/bart-large-cnn model for summarization. Readability analysis was performed using the textstat library. The frontend was developed using HTML5, CSS3, and JavaScript. Version control and collaboration were managed using Git and GitHub, with development carried out in Visual Studio Code.

4. System Design and Implementation

The system follows a three-tier architecture consisting of a client-side frontend, a Flask-based REST API layer, and a backend model inference layer. The BART model is loaded once during application startup to optimize performance. A /summarize POST endpoint accepts user input in JSON format, validates the text, applies token truncation for large inputs, and returns summarized output along with readability metrics.

The frontend interface provides a textarea for article input, a slider to control summary length, and a button to trigger summary generation using the Fetch API. The results section displays the generated summary, word counts, compression percentage, and readability level. A localStorage-based mechanism maintains a short history of recent summaries.

5. Key Features and Functionality

The application supports adjustable summary length, abstractive summarization using transformer models, readability evaluation through Flesch Reading Ease scores, compression ratio calculation, input validation, responsive user interface design, and a clearly defined JSON-based API for backend communication.

6. Technical Challenges and Solutions

One of the major challenges encountered during development was the large size of the BART model, which posed limitations for free-tier cloud deployment. This issue was mitigated by loading the model once at startup and using CPU-based inference. Very long text inputs were handled by applying a token truncation limit. Reproducibility was ensured through the use of a requirements.txt file and a clean virtual environment.

7. Results and Observations

Experimental evaluation showed that the BART model effectively condensed input text while retaining important contextual information. Compression ratios ranged from approximately 54 percent for short texts to nearly 85 percent for long articles. In several cases, the generated summaries exhibited improved readability due to shorter and clearer sentence structures.

8. Conclusion

The AI Writer with Text Summarization project successfully demonstrates the practical application of transformer-based NLP models in a real-world web application. The project provided hands-on experience in natural language processing, RESTful API development, frontend-backend integration, version control, and basic deployment considerations, making it a valuable learning experience.

References

Lewis et al. (2019). BART: Denoising Sequence-to-Sequence Pre-training. arXiv:1910.13461. Hugging Face Transformers Documentation. Flask Documentation. Flesch, R. (1948). A New Readability Yardstick.