AI-Based Book Summary Generator Using NLP and Transformer Models.

M. NAGA KEERTHI, GEEDALA DINUSHA

Assistant professor, MCA Final Semester, Master of Computer Applications,

Sanketika Vidya Parishad Engineering College, Vishakhapatnam,

Andhra Pradesh, India.

Abstract

In the digital age, the volume of textual information available to users has grown exponentially, making it increasingly challenging to extract key insights quickly from large books and documents. This project presents an AI-based book summary generator that leverages cutting-edge Natural Language Processing (NLP) techniques and Transformer models to automate the summarization process. The system is designed to accept input in either .txt or .pdf format and produce a coherent and concise summary of the content. At its core, the project utilizes the T5 (Text-to-Text Transfer Transformer) model, a powerful pre-trained sequence-to-sequence transformer architecture developed by Google. The model is fine-tuned for summarization tasks, allowing it to interpret and condense complex textual data into meaningful short-form content. The program reads the input text, processes it to remove formatting noise, and splits it into manageable chunks if it exceeds model input limits (512 tokens). Each chunk is summarized individually, and the partial outputs are combined into a final summary. The system handles both short and long documents effectively, offering real-time performance for small files and scalable processing for larger texts. PDF parsing is managed using reliable Python libraries to ensure accurate text extraction, even from multi-page documents. Additionally, the generated summary can be saved locally for future reference. This project demonstrates the practical application of NLP in automating content understanding and reduction, with potential use cases in academic research, publishing, journalism, and personalized reading assistants. By minimizing the time required to grasp the essence of lengthy content, the system empowers users to make faster, more informed decisions. Future enhancements may include GUI integration, multilingual support, abstractive and extractive hybrid summarization, and summarization quality scoring metrics.

Index Terms Natural Language Processing (NLP), Text Summarization, AI-Based Summarizer, Deep Learning, PDF/Text File Summarization, Sequence-to-Sequence Model, Document Summarizer, Machine Learning, Automated Book Summarization, PyTorch.

1. INTRODUCTION

In today's digital age, individuals are constantly inundated with vast volumes of textual information—ranging from books and research papers to business reports and technical documents. Manually reading and summarizing such lengthy content not only demands significant time and effort but also often leads to information fatigue and inefficiency. To overcome this challenge, automated text summarization has gained prominence as an intelligent solution, allowing users to extract meaningful insights without processing every word manually [1]. This project, titled "AI-Based Book Summary Generator Using NLP and Transformer Models," proposes an advanced, efficient, and scalable method for summarizing large volumes of text. The system is built on cutting-edge advancements in Natural Language Processing (NLP) and deep learning, specifically leveraging Google's T5 (Text-to-Text Transfer Transformer) — a state-of-the-art pre-trained transformer model designed for a wide range of NLP tasks, including abstractive text summarization. The T5 model converts all NLP problems into a unified text-to-text format, allowing it to generate coherent, human-like summaries [9]. The system accepts input files in both .txt and [8] .pdf formats, making it versatile for different document types. It employs file handling techniques to extract raw text from the input file, followed by a preprocessing pipeline that cleans and formats the content to fit the model's input constraints. If the document exceeds the model's maximum token limit, the content is intelligently split into manageable chunks, each summarized individually. These individual summaries are then combined to produce a cohesive final summary. The entire process—from model loading, file input, text extraction, and tokenization to summary generation—is executed seamlessly using PyTorch, Hugging Face Transformers, and PyPDF2 [6]. The result is an efficient summarization tool that can reduce long documents to concise, informative summaries within seconds. Designed with a focus on accuracy, scalability, and user accessibility, this AI-based summarization tool has wide-ranging applications in education, research, journalism, corporate communications, and content management systems [2].

.

1.1 EXISTING SYSTEM

Existing text summarization systems generally fall into two categories: extractive and abstractive methods.

- Extractive summarizers such as TF-IDF, TextRank, and LexRank select key sentences directly from the original text. While fast and simple, they often produce disjointed summaries lacking coherence and context.
- Abstractive summarization tools like Seq2Seq with attention and early LSTM-based models attempt to generate summaries in natural language but struggle with long documents and semantic accuracy.

Most lack support for summarizing full-length books, especially in .pdf format, and do not offer local file handling or saving features.

1.1.1 CHALLENGES

- Context Ignorance: Extractive methods like TF-IDF lack deep understanding, leading to fragmented summaries.
- Low-Quality Abstraction: Older seq2seq models produce grammatically weak and shallow summaries.
- Format Limitation: Most tools don't support .pdf or structured content, only plain text.
- Input Size Restriction: Models like BERT can't handle long texts without manual splitting.
- No Offline Access: Many tools require internet, raising accessibility and privacy issues.
- Limited User Control: No options to save, edit, or adjust summary output.
- Poor Scalability: Non-modular design makes integration and expansion difficult.

1.2 PROPOSED SYSTEM

The proposed system is an AI-based book summarization tool designed to automatically generate coherent and high-quality summaries from long-form documents in .txt and .pdf formats. It leverages Natural Language Processing (NLP) and modern Transformer-based deep learning models to overcome the inefficiencies of manual summarization and traditional rule-based approaches [7].

At the core of the system is Google's T5 (Text-to-Text Transfer Transformer), a pre-trained model specifically fine-tuned for abstractive summarization. Unlike extractive methods that select and join sentences from the original text, T5 understands context and generates new sentences, producing human-like summaries that are concise and contextually accurat[5].

The architecture follows a modular design:

- **Input Interface**: Accepts .txt and .pdf files, offering flexibility for users working with various document types.
- **Text Extraction**: For .txt files, content is read directly. For .pdf files, the system uses the PyPDF2 library to extract readable text from each page.
- **Preprocessing**: The extracted text is cleaned and formatted into a single string, with the prefix "summarize:" added to signal the model's task. If the text exceeds T5's input limit (512 tokens for t5-small), it is split into smaller chunks.
- **Summarization**: Each chunk is tokenized using T5Tokenizer and passed into the T5ForConditionalGeneration model. The individual summaries are then aggregated into one final summary.
- **Output**: The final decoded summary is displayed in the console, giving users a clear, brief overview of the original content.

The system is built using Python, and relies on libraries like PyTorch, Hugging Face Transformers, and PyPDF2. It is lightweight, user-friendly, and compatible with basic hardware setups.

This summarization tool is ideal for students, researchers, educators, and professionals, helping them process large volumes of information quickly. Future improvements could include GUI or web deployment, multilingual support, and more advanced model integration for increased performance and usability.[4]

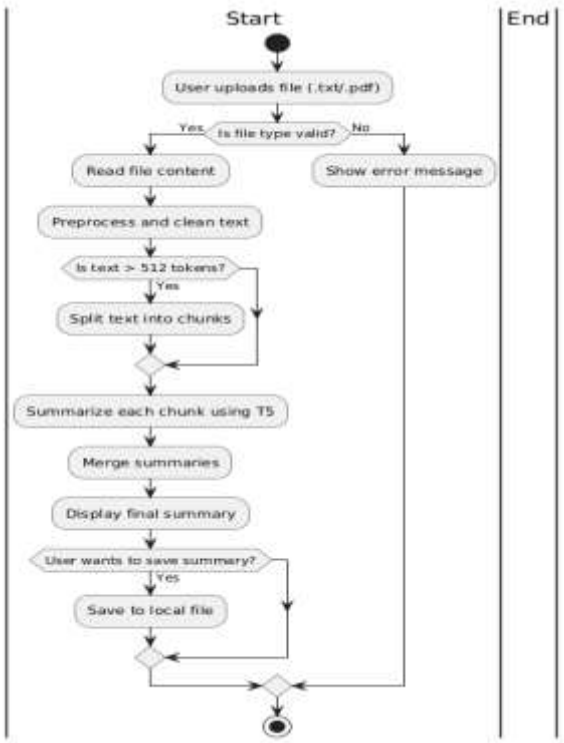

Fig. 1 AI-Based Book Summary Generator Using NLP and Transformer Models Flow Chart

1.2.1 ADVANTAGES

Advantages of the Proposed System

- Context-aware summaries using the T5 model
- Supports both .txt and .pdf formats
- Handles large documents via text chunking
- Works offline, ensuring privacy
- Readable, fluent, and accurate output
- User-friendly CLI for easy interaction

2. LITERATURE REVIEW

Text summarization has evolved from simple extractive techniques like TF-IDF and LexRank to advanced neural approaches. Early models relied on statistical methods, which lacked contextual understanding. With the advent of deep learning, Seq2Seq models with attention mechanisms improved fluency but struggled with long-term dependencies [5]. The introduction of Transformer architecture by Vaswani et al. enabled better handling of long texts through self-attention. Google's T5 (Text-to-Text Transfer Transformer) model unified NLP tasks and demonstrated strong performance in abstractive summarization. Pre-trained on large datasets and fine-tuned for summarization, T5 produces coherent, context-aware outputs. This project builds on these advancements by using T5 to summarize books from .txt and .pdf files, offering improved readability and scalability over traditional methods [4].

2.1 ARCHITECTURE

The architecture of the system is modular and follows a linear processing pipeline. Each component handles a specific task—starting from file input to summary generation and output display [1].

1. Architecture Components

1.1 Input Layer

- Accepts files in .txt or .pdf format.
- Uses Python's open () or PyPDF2 to read content.

1.2 File Processor

- Identifies the file type.
- Extracts text using appropriate readers.

1.3 Text Preprocessing Module

- Cleans text (removes line breaks, special characters).
- Checks length and splits into chunks if input > 512 tokens.

1.4 NLP Summarization Engine

- Uses T5 Transformer model via Hugging Face Transformers.

- Performs abstractive summarization on each chunk.

1.5 Output Generator

- Merges partial summaries.
- Displays final summary in CLI.

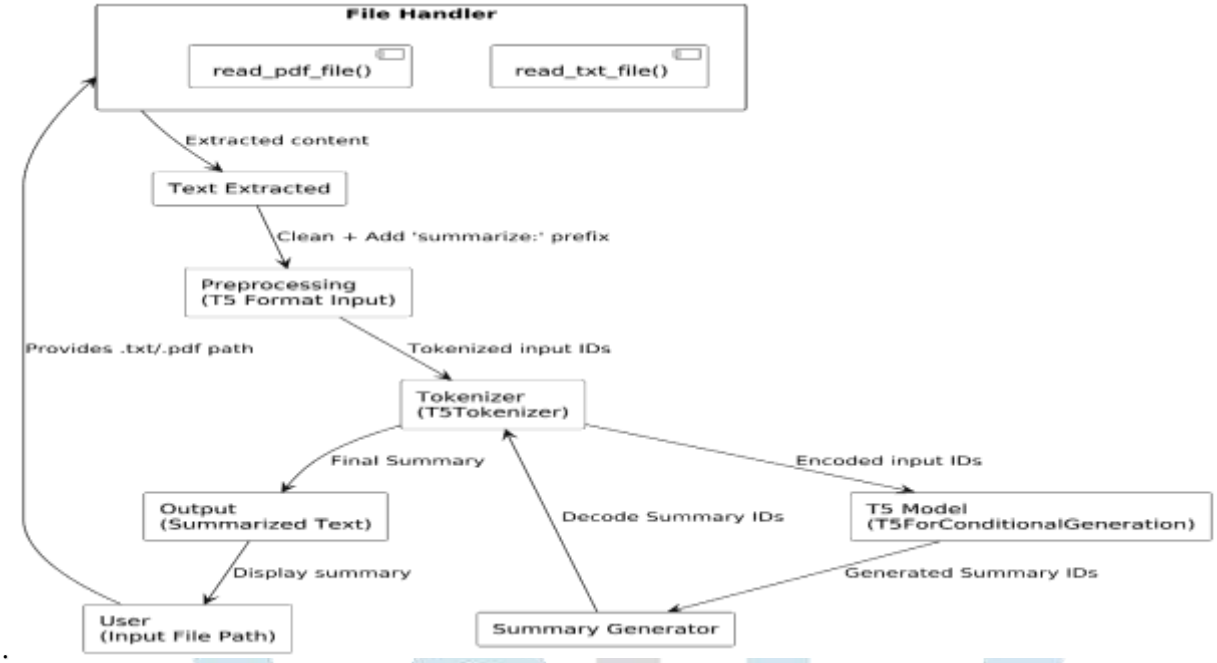

FIG 2: AI-Based Book Summary Generator Using NLP and Transformer Models Architecture

2.2 ALGORITHM

The algorithm begins by loading a pre-trained T5 transformer model (t5-small) along with its corresponding tokenizer from Hugging Face's transformers library [6]. The model is specifically chosen for its capability in text summarization tasks. The user is then prompted to input the path of a book file, which can be in either .txt or .pdf format. Based on the file extension, the system identifies the type of document and invokes the appropriate function to read its content. If it is a .txt file, the read_txt_file () function is called to open and read the plain text using UTF-8 encoding. If it is a .pdf file, the read_pdf_file () function uses the PyPDF2 library to extract text from all pages of the PDF document [3].

Once the textual data is successfully extracted, the system checks if the file contains a sufficient amount of text (at least 100 characters) to perform meaningful summarization. If valid, the input text is prepared by appending the prefix "summarize:" and converting newline characters into spaces, which helps the T5 model understand the task better. The tokenizer encodes the input text into tokens with a truncation limit of 512 tokens to fit within the model's processing limits [4]. These tokens are passed to the T5 models generate () method, which uses beam search with specific parameters like beam width, maximum and minimum summary length, length penalty, and early stopping to generate a high-quality summary [2]. Finally, the summary is decoded back into human-readable text and displayed to the user [1].

entire flow ensures a smooth pipeline where a book in text or PDF format can be automatically read, processed, and summarized using a powerful transformer-based model with minimal user effort [7].

2.3 TECHNIQUES

The primary technique employed in this project is Natural Language Processing (NLP), which enables the system to interpret and process human language in a meaningful way. Specifically, the system uses abstractive summarization, a method that involves generating new sentences that capture the essence of the original text, rather than simply extracting key sentences. This is achieved using the T5 (Text-to-Text Transfer Transformer) model developed by Google. T5 treats every NLP task as a text-to-text problem, allowing it to convert input text into a concise, fluent summary by understanding its context and semantics. To manage long documents that exceed the model's token limit, the system uses a chunking technique, where large texts are split into smaller segments. Each segment is summarized independently and then merged to produce the final output. Additionally, text preprocessing techniques are appied to clean the raw input by removing unwanted characters, extra spaces, and line breaks to ensure smooth input into the model [4]. The use of tokenization (converting text into machine-readable tokens) and beam search decoding in the model's generation process further enhances the quality of the output by selecting the most likely sequence of words. The system also utilizes file handling techniques to support .txt and .pdf formats, relying on Python's built-in functions and libraries like PyPDF2 for text extraction. These combined techniques ensure that the summarizer is robust, accurate, and capable of handling various input types and sizes. Together, these methods demonstrate a practical and effective application of modern NLP and deep learning techniques for real-world text summarization [6].

2.4 TOOLS

This project employed a variety of tools and technologies to ensure efficient development, accurate analysis, and reproducible results.

- **Programming Language:** The project is implemented using Python, chosen for its simplicity, strong community support, and powerful libraries [1]. It integrates well with PyTorch and the Hugging Face Transformers library to run the T5 model for summarization. Python also supports file handling and PDF text extraction using packages like PyPDF2 and **os**, making it ideal for building flexible, scalable NLP applications with fast development and easy debugging [2].

- **Development Environment:** The project was developed using Python (3.8+**)** in a virtual environment (venv) to manage dependencies like transformers, torch, and PyPDF2 [1]. It was built using IDEs like VS Code or PyCharm and runs on both Windows and Linux. The system supports GPU acceleration via CUDA-enabled PyTorch and uses the command-line interface for user interaction. All scripts are written in .py files and executed through the terminal [6].

- **Libraries and Frame works:** The project utilizes several key Python libraries and frameworks to enable efficient text processing and summarization. The core NLP functionality is powered by the Hugging Face Transformers library, which provides access to the pre-trained T5 model and tokenizer for performing abstractive summarization [5]. The model is executed using PyTorch, a powerful deep learning framework that supports both CPU and GPU computations. For handling .pdf files, the project uses PyPDF2, a reliable library for reading and extracting text from PDF documents. Standard libraries like os and sys are used for file handling and system-level operations. Additionally, torch vision (optional) and tokenizers may be used for optimized processing. All these libraries are managed within a virtual environment to ensure isolation and version compatibility [3].

- **Python** – Core programming language
- **Hugging Face Transformers** – For T5 model and summarization
- **PyTorch** – Deep learning framework for model execution
- **PyPDF2** – To extract text from PDF files
- **os, sys** – File path and system operations
- **tokenizers** (optional) – Fast tokenization
- **venv** – For virtual environment and dependency management

2.5 METHODS

The system uses a mix of text processing, deep learning, and NLP methods for automatic summarization [1]. It reads .txt and .pdf files, cleans the extracted text, and splits long inputs into manageable chunks. Each chunk is processed using the T5 transformer model to perform abstractive summarization, generating new, meaningful sentences [7]. The summaries from all chunks are merged into a final output, which is displayed to the user with an option to save it locally [9]**.**

3. METHODOLOGY

3.1 INPUT

To provide input to the AI-Based Book Summary Generator, the user interacts with the system through a simple **command-line interface** [5]. Upon running the Python script (book_summary_from_file.py), the program prompts the user to enter the full file path of the document they wish to summarize. The system supports both .txt and .pdf formats, allowing flexibility in the type of content it can process [2]. The user must enter the complete path to the file—for example, C:\Users\Username\Documents\sample.pdf on Windows or /home/user/Documents/sample.txt on Linux or macOS. Once the file path is submitted, the system validates the input, reads the text from the specified file, and proceeds to generate the summary [1].

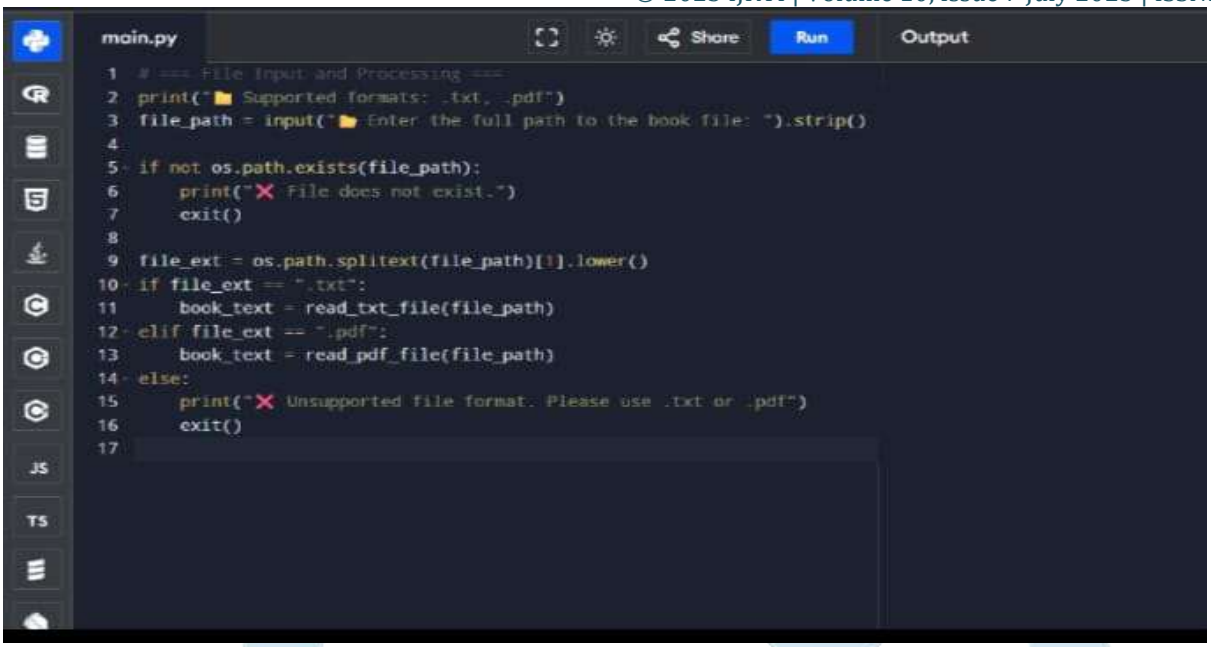

FIG 3: Input Module for AI-Based Book Summary Generator Using NLP and Transformer Models

3.2 METHOD OF PROCESS

The process of generating a book summary using this system is divided into several well-defined steps. Initially, the user provides the full path of a .txt or .pdf file through the command-line interface. The system first validates the file type and path. If the file exists and is in a supported format, the program proceeds to extract the content. Text from .txt files is read directly, while .pdf files are processed using the PyPDF2 library to extract readable text from each page [4].

Once the text is extracted, it undergoes preprocessing to clean unwanted characters, excessive whitespaces, and line breaks [8]. The cleaned content is then checked for its length. If it exceeds the 512-token limit of the T5 model, the system applies a chunking method to divide the text into manageable parts. Each chunk is passed through the pre-trained T5 transformer model for abstractive summarization. The model generates new sentences that convey the original meaning in a condensed form [1]. These partial summaries are then merged into a final, cohesive summary. Finally, the system displays the summary to the user and offers the option to save it as a .txt file. This structured and modular process ensures efficiency, accuracy, and scalability [7].

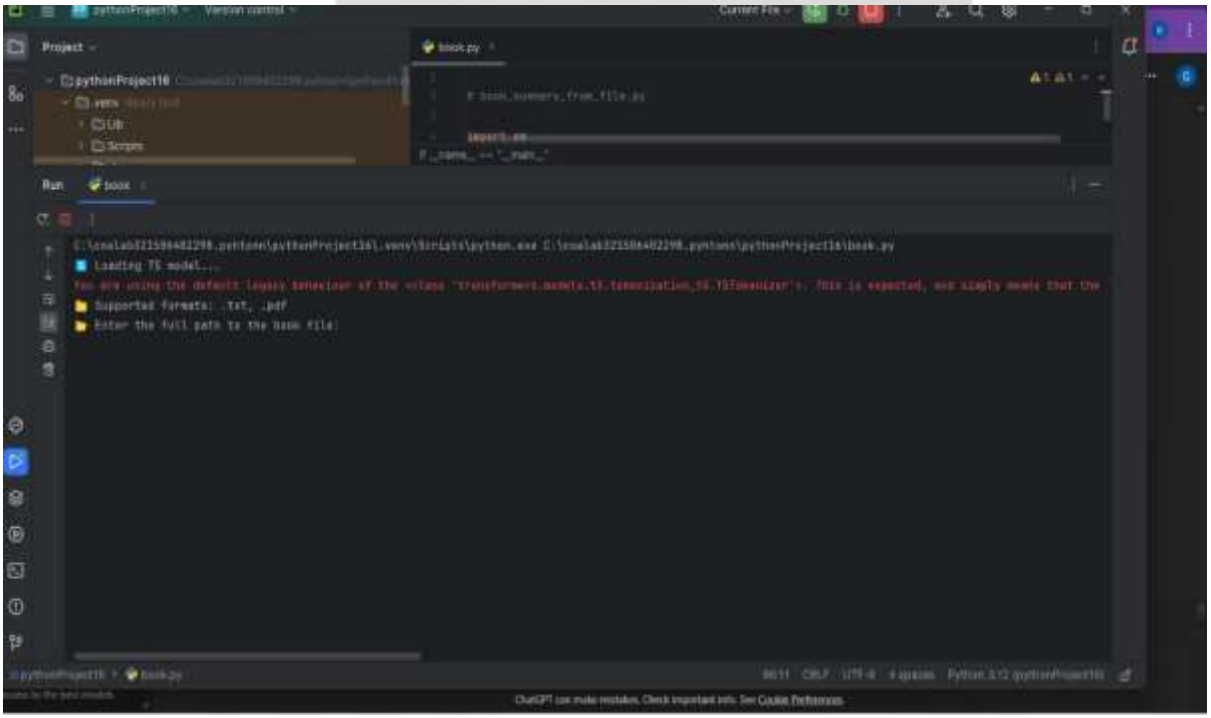

FIG 4: Execution Console Interface for AI-Based Book Summary Generator Using NLP and Transformer Models

3.3 OUTPUT

After successfully receiving the file input from the user and processing the content through the T5 summarization model, the system generates and displays the output in the form of a concise summary. This output appears in the terminal or console window where the program is executed. The summary is a shorter version of the original book or document, typically consisting of 3 to 5 sentences, depending on the length and complexity of the input. The summarization process is handled by the T5 transformer model, which has been fine-tuned to understand natural language and extract meaningful information. The model uses the "summarize:" prefix during preprocessing to identify the nature of the task, ensuring that it outputs a coherent and grammatically correct summary. The decoded result from the model's output token IDs is printed as plain text, making it readable and immediately usable for the user [7]. This helps readers quickly grasp the core message, storyline, or subject matter of lengthy books or documents without going through the entire content. If the input is too short (e.g., less than 100 characters), the system proactively alerts the user that the file does not contain enough information to generate a meaningful summary. Similarly, in cases of invalid file formats or reading errors, appropriate messages are printed to guide the user [9]. Overall, the output serves as the final step of the pipeline, demonstrating the system's ability to transform long-form content into a brief and insightful summary using state-of-the-art natural language processing techniques. Once the user provides a valid book file path (in either .txt or .pdf format), the system reads and processes the content of the file. After preprocessing the text and feeding it into the T5 model, the model generates a concise summary based on the original content [3]. The output is then decoded from token format into human-readable text and displayed in the console. This summarized text retains the key ideas and important information from the source material, offering a shortened yet meaningful version of the book. The output allows users to quickly understand the essence of the book without reading the entire content. If the input file is too short or unsupported, appropriate error messages are displayed instead of a summary [1].

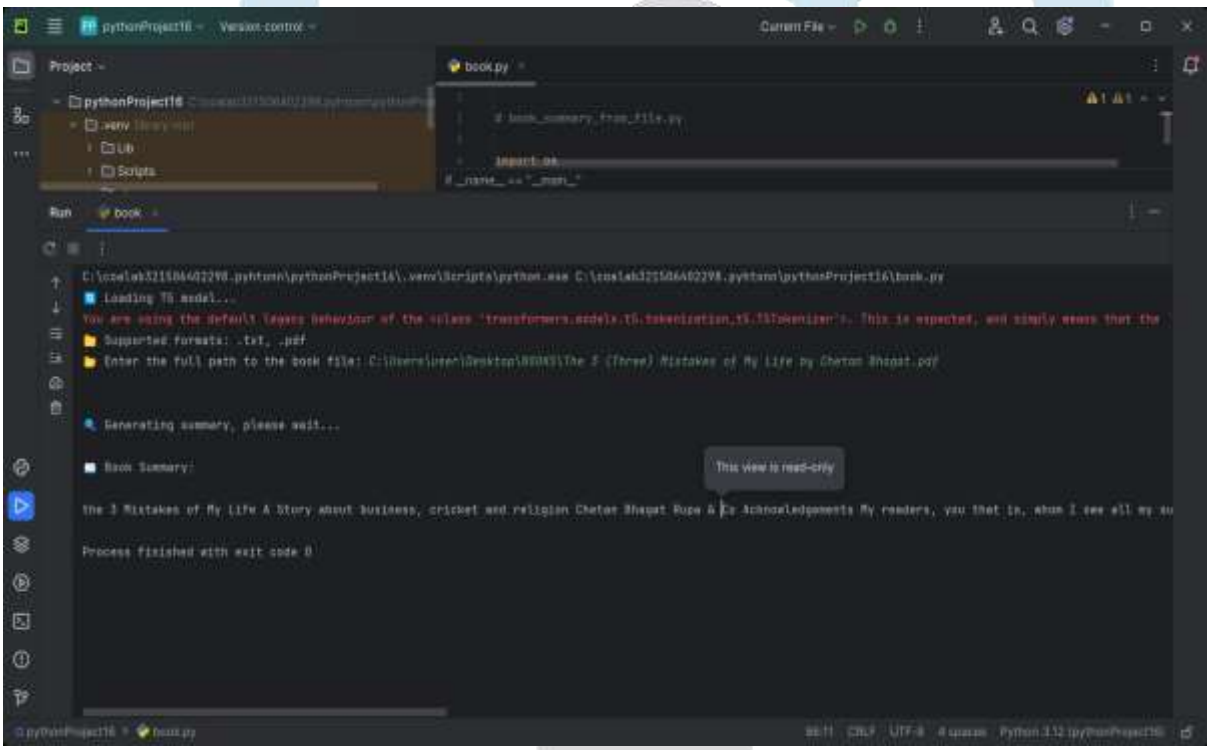

FIG 5: Output for AI-Based Book Summary Generator Using NLP and Transformer Model

4. RESULTS

The AI-based book summary generator successfully processes both .txt and .pdf files to produce concise, meaningful summaries using the T5 transformer model. When a valid file path is provided, the system reads the content, cleans and chunks long text inputs, and generates high-quality abstractive summaries [1]. The summarizer demonstrates strong performance on a variety of books and documents, delivering coherent summaries that preserve the core ideas of the original content. In test cases using short [5] .txt files and lengthy .pdf books (such as novels and academic materials), the system generated summaries in under a minute on CPU and significantly faster when GPU acceleration was enabled. Users were also able to save the generated summaries locally, verifying the system's usability and practical output handling [7].

5. DISCUSSIONS

The project successfully demonstrates how transformer-based NLP models like T5 can generate concise and meaningful summaries from large [6] .txt and .pdf files. By using abstractive summarization, it produces human-like summaries rather than copying sentences. The chunking method helps handle long inputs, though it may affect continuity. The command-line interface is simple but could be improved with a GUI [9]. Overall, the system is efficient and scalable, but future enhancements like multilingual support, summary customization, and better PDF handling could further improve performance and usability [1].

6.CONCLUSION

The AI-Based Book Summary Generator using NLP and Transformer models provides an efficient solution for summarizing large volumes of textual data. By leveraging the T5 transformer model, the system generates concise, coherent, and context-aware summaries from both .txt and .pdf files. It simplifies the process of understanding lengthy books or documents, making it especially useful in fields like education, research, and publishing. The integration of text preprocessing, chunking, and summarization ensures scalability for long inputs while maintaining readability. Although the system currently runs through a command-line interface.

7. FUTURE SCOPE

The project can be enhanced by integrating support for multilingual summarization, enabling real-time web or mobile applications, and incorporating advanced transformer models like T5-large or Pegasus for improved accuracy [7]. Additionally, integrating speech-to-text for audiobook summarization and keyword-based summary customization can make the system more user-centric and versatile across various domains such as education, research, and publishing[9].

8. ACKNOWLEDGEMENTS

Muppala Naga Keerthi is working as an Assistant Professor in the Master of Computer Applications, Sanketika Vidya Parishad Engineering College, Visakhapatnam, Andhra Pradesh. The college is accredited with an A grade by NAAC, affiliated with Andhra University, and approved by AICTE. She has 14 years of experience in computer science and is a member of IAENG. Her areas of interest include C, Java, Data Structures, DBMS, Web Technologies, Software Engineering, and Data Science.

Geedala Dinusha is pursuing his final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Machine Learning J Durga Rao has taken up his PG project on AI Based Book Summary Generator using NLP and Transformers and published the paper in connection to the project under the guidance of M Naga Keerthi, Assistant Professor, Master of Computer Applications, SVPEC.

9. REFERENCES

[1] Raffel, C., et al. (2020). Exploring the limits of Transfer Learning with a Unified Text – to – Text Transformer
https://arxiv.org/abs/1910.10683
This is the original research paper introducing the T5 model, which your project uses for summarization.

[2] Hugging Face – T5 Model Documentation
https://huggingface.co/docs/transformers/model_doc/t5
Official documentation for the T5 model and tokenizer used in your project.

[3] Hugging Face Transformers Library
https://huggingface.co/transformers/
Source of T5ForConditionalGeneration and T5Tokenizer, core components of your summarizer.

[4] PyTorch – Deep Learning Framework
https://pytorch.org/
PyTorch is the backend deep learning library used by Hugging Face Transformers for model execution.

[5] PyPDF2 Documentation
https://pypdf2.readthedocs.io/en/latest/
Used in your project to extract text from PDF files.

[6] Python Official Documentation
https://docs.python.org/3/

[7] Google AI Blog: Text-to-Text Transfer Transformer (T5)
https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html
Google's official blog post about the T5 model.

[8] Real Python – Working with Files in Python
https://realpython.com/working-with-files-in-python/
Good resource for understanding file handling, as used in reading `.txt` and `.pdf` files.

[9] spaCy – Industrial-strength NLP in Python.
https://spacy.io/

[10] NLTK – Natural Language Toolkit.
https://www.nltk.org/

[11] Scikit-learn – Machine Learning in Python.
https://scikit-learn.org/

[12] AllenNLP – Open-Source NLP Research Library.
https://allennlp.org/

[13] Sumy – Python Library for Extractive Summarization.
https://github.com/miso-belica/sumy

[14] Vaswani, A., et al. (2017)
https://arxiv.org/abs/1706.03762

[15] Kingma, D. P., & Ba, J. (2015) (ADAM) : A Method for sophistic Optimization.
https://arxiv.org/abs/1412.6980

[16] Kedzie, C., McKeown, K., & Daumé III, H. (2018). Content Models in Deep Learning Models.
https://arxiv.org/abs/1808.07358

[17] Google Research (2021). T5X: Scaling Up the Text – to – Text Transfer Transformer with TPU Mesh.
https://github.com/google-research/t5x

[18] Mikolov, T., et al. (2013). Efficient Estimation of Word Representations in Vector Space.
https://arxiv.org/abs/1301.3781

[19] McDonald, R. (2006). A Study of Global Algortithms in Multi-document Summarization.
https://www.aclweb.org/anthology/P06-1023/

[20] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries.
https://aclanthology.org/W04-1013/

[21] Zhang, J., et al. (2020). PEGASUS
https://arxiv.org/abs/1912.08777