Text Summarization using NLP, Base T5 Transformer, and Streamlit Application

Version: 1.0

Date: May 14, 2025

Author: [Your Name]

Institution: [Your University/Institution Name]

Department: [Your Department Name]

Table of Contents:

1. Introduction

1.1. Project Overview

1.2. Motivation and Problem Statement

1.3. Objectives

1.4. Scope of the Project

1.5. Target Audience for this Documentation

2. Background and Literature Review

2.1. Natural Language Processing (NLP)

2.1.1. Overview of NLP

2.1.2. Key NLP Tasks Relevant to Summarization

2.2. Text Summarization

2.2.1. Introduction to Text Summarization

2.2.2. Types of Text Summarization (Extractive vs. Abstractive)

2.2.3. Challenges in Text Summarization

2.3. Transformer Networks

2.3.1. Introduction to Transformer Architecture

2.3.2. Self-Attention Mechanism

2.3.3. Encoder-Decoder Structure

2.4. T5 Transformer Model

2.4.1. Overview of T5

2.4.2. "Text-to-Text" Framework

2.4.3. Advantages of Using T5 for Summarization

2.5. Streamlit

2.5.1. Introduction to Streamlit

2.5.2. Advantages for Rapid Application Development

2.5.3. Use Case in NLP Applications

2.6. Relevant Prior Work and Research

2.6.1. Existing Text Summarization Techniques

2.6.2. Applications of Transformer Models in Summarization

2.6.3. Tools and Libraries Used in Similar Projects

3. Methodology and System Design

3.1. Overall System Architecture

3.1.1. Block Diagram of the System

3.1.2. Data Flow within the System

3.2. Data Acquisition and Preprocessing

3.2.1. Data Sources (e.g., Datasets used for training/testing - Based on the code, the model uses a pre-trained T5 base model, so training/fine-tuning datasets are not explicitly required for basic functionality, but would be for fine-tuning.)

3.2.2. Data Cleaning Techniques (e.g., removing special characters, HTML tags - Implemented in preprocess\_text function)

3.2.3. Text Preprocessing Steps (e.g., tokenization, lowercasing - Handled by the T5 tokenizer internally, but also basic cleaning in preprocess\_text)

3.3. Model Implementation

3.3.1. Choice of Base T5 Model and Justification (The code uses t5-small. Discuss why t5-small was chosen, e.g., for faster inference, resource constraints, ease of deployment in a web app, while acknowledging t5-base or larger models could offer better performance.)

3.3.2. Library Used for T5 Implementation (transformers library)

3.3.3. Fine-tuning Process (if applicable - The provided code uses a pre-trained model. If you fine-tuned, describe that process here.)

3.3.3.1. Dataset Used for Fine-tuning (If applicable)

3.3.3.2. Training Parameters (e.g., learning rate, batch size, epochs - If applicable)

3.3.3.3. Evaluation Metrics during Training (If applicable)

3.3.4. Model Saving and Loading (Models are loaded using from\_pretrained and cached with @st.cache\_resource.)

3.4. Streamlit Application Development

3.4.1. User Interface Design (Describe the layout: file uploader, URL input, text area, audience selection, summarization button, study material generation, visualization options, output areas for summaries and study materials.)

3.4.1.1. Input Area for Text (Includes direct text input, file upload for PDF, DOCX, PPTX, TXT, and URL/YouTube input)

3.4.1.2. Button for Summarization and Study Material Generation

3.4.1.3. Display Area for the Summary and Study Materials (Organized using Streamlit expanders and tabs)

3.4.1.4. Potential for Parameter Tuning (e.g., audience selection)

3.4.2. Integration of the T5 Model with Streamlit (The load\_models function loads models and @st.cache\_resource ensures they are loaded only once. The generate\_summary function is called when the "Summarize" button is clicked.)

3.4.3. Deployment Considerations (Local Execution and potential for wider deployment)

4. Implementation Details (Referring to the Code)

4.1. Python Libraries Used

streamlit: Used for building the interactive web user interface.

os: For interacting with the operating system, primarily for handling file paths and temporary files.

tempfile: For creating and managing temporary files, essential for processing uploaded files and extracted audio/video.

random: Used for generating random elements, specifically for creating analogies in child-friendly summaries and varying study questions.

typing: Provides type hints for improved code readability and maintainability.

transformers: Core library for loading and utilizing the T5 and BART transformer models and their tokenizers.

nltk: Natural Language Toolkit, used for text processing tasks, specifically forpunkt (tokenizer) and stopwords (for TF-IDF and word clouds).

spacy: Advanced NLP library used for tasks like Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and extracting noun chunks, which are used in generating study materials and the entity network visualization.

re: Python's regular expression module, used for text cleaning and pattern matching in preprocessing and URL parsing.

collections.defaultdict: While imported, it is not explicitly used in the provided code snippet's main logic but is a useful data structure for potential future extensions.

sklearn.feature\_extraction.text.TfidfVectorizer: Used for calculating TF-IDF scores to identify key terms, specifically for the researcher summary.

numpy: Used for numerical operations, particularly with TF-IDF calculations and sorting.

PyPDF2.PdfReader: Used for extracting text content from uploaded PDF files.

docx.Document: Used for extracting text content from uploaded DOCX files.

pptx.Presentation: Used for extracting text content from uploaded PPTX files.

networkx: Used for creating and manipulating the graph data structure for the entity relationship network visualization.

matplotlib.pyplot: Used as the backend for creating the static word cloud visualization.

plotly.graph\_objects: Used for creating the interactive entity relationship network visualization.

requests: Used for making HTTP requests to fetch content from web URLs.

bs4.BeautifulSoup: Used for parsing HTML and XML content fetched from web pages to extract readable text.

youtube\_transcript\_api.YouTubeTranscriptApi: Used for fetching transcripts from YouTube videos. Includes handling for NoTranscriptFound and TranscriptsDisabled exceptions.

moviepy.editor.VideoFileClip: Used for loading and manipulating video files, specifically for extracting the audio track.

whisper: OpenAI's audio transcription model, used for transcribing audio extracted from video files.

concurrent.futures: Used for potentially running tasks in parallel, although its explicit use for parallel summary generation wasn't shown in the snippet, the import suggests this capability could be added or used elsewhere.

4.2. Code Structure and Organization

The provided code is structured into logical sections using comments:

Import necessary libraries: Lists all external libraries used.

Global Variables & Configuration: Includes NLTK download attempt and Streamlit page configuration.

Custom CSS Styling: Defines the visual theme of the Streamlit application.

CORE MODEL LOADING: Contains the load\_models function, which uses @st.cache\_resource to efficiently load the T5, BART (for child summaries), spaCy, and Whisper models.

FILE AND CONTENT PROCESSING UTILITIES: Groups functions for handling different input types: extract\_text\_from\_file for documents, transcribe\_audio\_with\_whisper and extract\_audio\_from\_video for media, and Workspace\_and\_parse\_url for web content. It also includes the generic preprocess\_text function.

SUMMARY GENERATION: Contains the core generate\_summary function, which orchestrates the summarization based on the selected audience and uses either the T5 or BART model.

STUDY MATERIAL GENERATION: Includes the generate\_study\_materials function, which leverages spaCy to extract key phrases and generate flashcards, questions, and diagram suggestions.

VISUALIZATION UTILITIES: Contains functions for creating visualizations: create\_wordcloud and create\_entity\_network.

(Note: The provided code snippet appears to end mid-function for create\_entity\_network. The actual application logic for calling these functions and displaying the Streamlit UI elements would follow these utility functions.)

4.3. Detailed Explanation of Key Code Snippets (Illustrative - based on provided code)

Model Loading (load\_models): This function is critical as it loads all the necessary NLP models. The @st.cache\_resource decorator is used to ensure these large models are loaded into memory only once when the application starts, significantly improving performance for subsequent user interactions. It loads t5-small for general summarization, facebook/bart-large-cnn specifically fine-tuned for summarization and potentially better for the child audience, a small English spaCy model (en\_core\_web\_sm) for linguistic processing, and the base Whisper model for audio transcription. Error handling is included to stop the app if model loading fails.

Text Preprocessing (preprocess\_text): This function handles basic text cleaning, such as replacing multiple spaces and stripping whitespace. It also includes a mode parameter to apply audience-specific rules, like removing potentially complex characters or very long words for the "child" audience.

Summary Generation (generate\_summary): This is the core summarization logic. It takes the input text and the target audience as parameters. It uses a conditional structure to select the appropriate model (child\_summarizer which uses BART for the "child" audience, and the T5 model for others). For T5, it includes logic to handle potentially long input texts by splitting them into chunks and summarizing each chunk, then joining the summaries. It also configures generation parameters (e.g., max\_length, min\_length, num\_beams, temperature) based on the audience to tailor the summary output. For the "researcher" audience, it attempts to extract key terms using TF-IDF.

Handling Different Input Types: The code includes dedicated functions to process various input sources:

extract\_text\_from\_file: Supports PDF, DOCX, PPTX, and TXT files using libraries like PyPDF2, python-docx, and python-pptx.

Workspace\_and\_parse\_url: Handles fetching content from general web URLs using requests and BeautifulSoup and specifically fetches transcripts from YouTube URLs using youtube\_transcript\_api.

extract\_audio\_from\_video: Extracts audio from video files using moviepy.

transcribe\_audio\_with\_whisper: Transcribes the extracted audio using the Whisper model. These functions ensure the application can process diverse content types.

Study Material Generation (generate\_study\_materials): This function takes the processed text and the spaCy model. It identifies key phrases using noun chunks and generates a list of flashcards, practice questions, diagram suggestions, and a simple study plan based on these key phrases. Randomization is used to create varied questions and analogies (in the child summary).

Visualization Utilities:

create\_wordcloud: Generates a word cloud image using the wordcloud library and NLTK stopwords.

create\_entity\_network: Creates an interactive graph visualization using networkx and plotly. It identifies entities using spaCy and draws connections between entities that appear in the same sentence, with edge weight potentially indicating co-occurrence frequency. Node size can be influenced by the length of the entity text, and node colors are based on entity type.

```

5. Evaluation and Results

5.1. Evaluation Metrics

5.1.1. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores (ROUGE-1, ROUGE-2, ROUGE-L) - These would typically be calculated offline on a test dataset with human-generated reference summaries.

5.1.2. BLEU (Bilingual Evaluation Understudy) score (if applicable) - Less common for summarization than ROUGE, but can be used.

5.1.3. Other relevant metrics (e.g., manual evaluation for coherence, readability, factuality).

5.2. Experimental Setup

5.2.1. Datasets used for evaluation.

5.2.2. Comparison with baseline models or existing summarization techniques (if applicable).

5.2.3. Hyperparameter tuning and its impact on performance (if applicable).

5.3. Results and Analysis

5.3.1. Presentation of evaluation results (tables, graphs).

5.3.2. Interpretation of the results and discussion of the model's performance.

5.3.3. Qualitative analysis of the generated summaries (examples of good and bad summaries).

5.3.4. Error analysis and identification of limitations.

6. Discussion and Conclusion

6.1. Summary of Findings

6.2. Strengths of the Developed System (e.g., multiple input types, audience-specific summaries, study material generation, visualizations)

6.3. Limitations of the Developed System (e.g., performance with very long texts due to chunking, accuracy of child summaries, potential issues with specific file types or website structures, reliance on transcript availability for YouTube, resource usage of models)

6.4. Potential Future Work

6.4.1. Exploring different T5 model variations (e.g., t5-base, t5-large) or other state-of-the-art summarization models.

6.4.2. Implementing more sophisticated text preprocessing and chunking strategies for handling long documents.

6.4.3. Incorporating techniques for evaluating summary quality more robustly (e.g., using factual consistency metrics).

6.4.4. Deploying the application on a cloud platform for wider access.

6.4.5. Adding more advanced user interface features and options (e.g., controlling summary length with a slider, choosing specific sections to summarize).

6.4.6. Enhancing study material generation (e.g., generating multiple choice questions, providing answers).

6.4.7. Improving the entity network visualization (e.g., showing edge labels indicating relationship types).

6.5. Conclusion

7. References

List of all academic papers, articles, books, and online resources cited in the documentation. Follow a consistent citation style (e.g., APA, MLA). Include references for:

The T5 model and its original paper.

The BART model (if used for child summaries) and its original paper.

The Transformer architecture paper ("Attention Is All You Need").

The Streamlit library.

Other libraries used (e.g., spaCy, Hugging Face Transformers).

Any datasets used for evaluation or fine-tuning.

Relevant literature on text summarization techniques.

8. Appendices (Optional)

A. Full Source Code Listing

B. Detailed Experimental Results and Evaluation Outputs

C. Glossary of Terms

D. Screenshots of the Application Interface