

Automated Text Summarization from YouTube Transcripts

An NLP Project Report

Pallav (2201135)
Sumit Raj (2201209)

Sumit (2201207)
Ujjwal (2201217)

November 8, 2025

1 Introduction

The proliferation of educational, informational, and entertainment content on platforms like YouTube has created an information overload problem. Manually watching long videos to extract key information is time-consuming and inefficient. Automatic text summarization, a key application of Natural Language Processing (NLP), offers a powerful solution by condensing a source text into a shorter version while preserving its core meaning.

This project presents a self-hosted Flask web application designed to summarize YouTube videos. Users can simply provide a video URL, and the system performs the following key functions:

- Fetches the video's full transcript automatically.
- Processes the text to generate a concise summary.
- Offers both extractive and abstractive summarization methods to suit different user needs.

The core objective is to provide an efficient, user-friendly tool that saves time and enhances learning by providing quick access to the essential information contained in video content.

2 System Architecture and Workflow

The application is designed with a classic three-tier architecture, separating the user interface, business logic, and data processing. This modular design ensures that the system is scalable and easy to maintain. The overall workflow is depicted below and follows a clear, sequential process from user input to summary output.

2.1 Architectural Diagram

The data flow begins with the user interacting with the web interface and culminates in the display of the generated summary.

2.2 Workflow Steps

1. **User Input:** The user pastes a YouTube video URL into the frontend web form.
2. **Request Handling:** The Flask backend receives the URL via a POST request to the `/summarize` API endpoint.

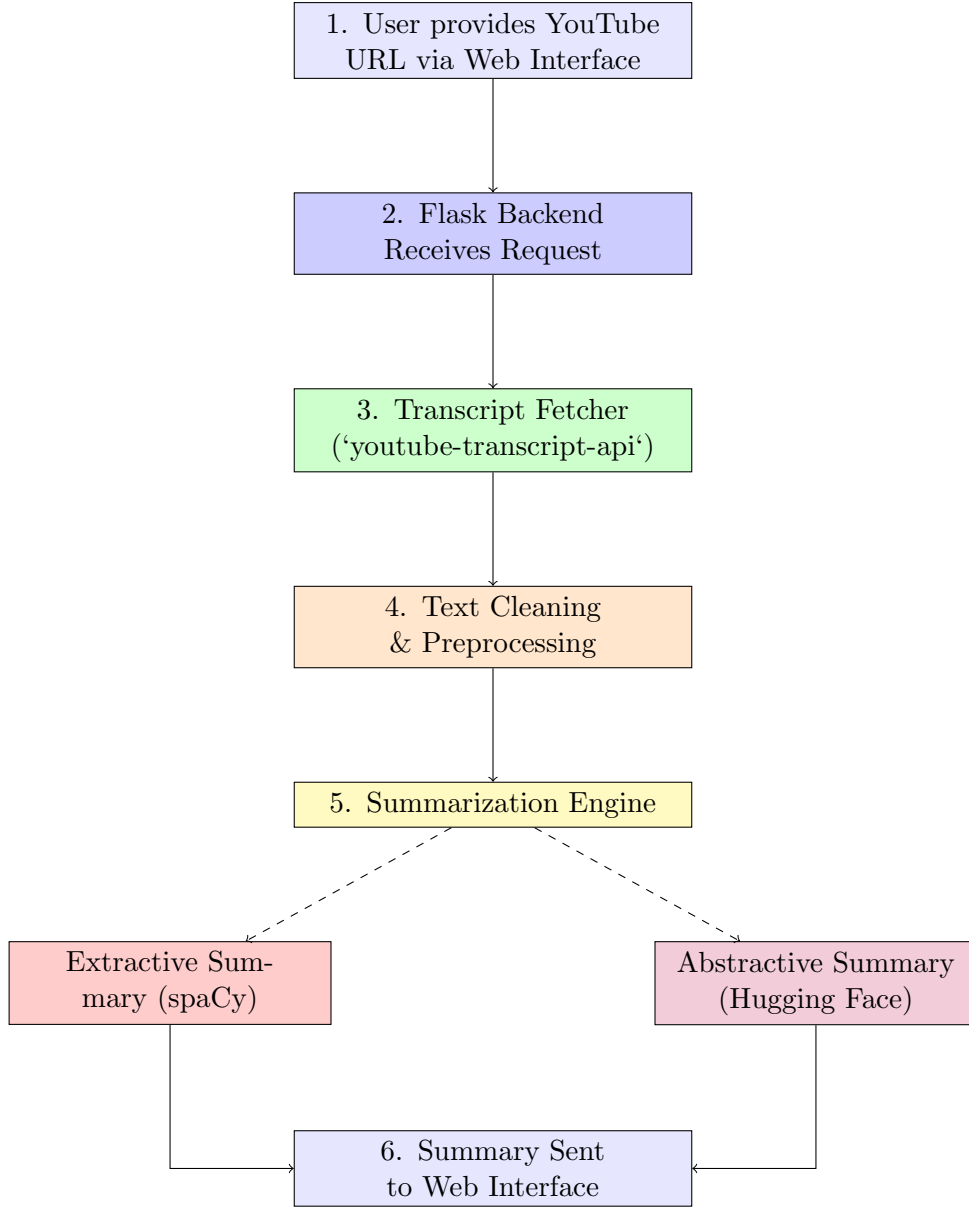


Figure 1: System Workflow Diagram

3. **Transcript Fetching:** The video ID is extracted from the URL. A dedicated module then calls the 'youtube-transcript-api' to retrieve the full transcript.
4. **Text Processing:** The raw transcript text is combined into a single document and cleaned of any residual HTML entities.
5. **Summarization:** The cleaned text is passed to the selected summarization model.
 - **Extractive Method:** This is the default and primary method. It analyzes the text to identify and extract the most significant sentences.
 - **Abstractive Method:** If selected, this method uses a pre-trained model to generate a new summary that captures the original text's meaning. The system includes a robust fallback to the extractive method if the abstractive model fails.
6. **Display Output:** The final summary, along with compression statistics, is sent back to the user's browser as a JSON response and displayed on the webpage.

3 Implementation Details

3.1 Backend Technology

The server-side logic is developed entirely in Python, utilizing the **Flask** micro-framework. Flask was chosen for its lightweight nature and simplicity, making it ideal for creating the RESTful API that powers the application.

3.2 Extractive Summarization

The extractive summarization functionality is the core of this project and is implemented using the **spaCy** library, a powerful tool for industrial-strength NLP. The algorithm operates as follows:

1. The input text is parsed into a spaCy ‘doc’ object, which segments the text into sentences and words.
2. A frequency map of all non-stop words is created. Stop words (like “the,” “is,” “a”) are ignored as they do not carry significant meaning.
3. Each word’s frequency is normalized by dividing it by the frequency of the most common word.
4. Each sentence is assigned a score based on the sum of the normalized frequencies of its constituent words.
5. The sentences with the highest scores are selected to form the final summary, which are then presented in their original order to maintain coherence.

This frequency-based approach is computationally efficient and reliably extracts the main points of a document.

3.3 Abstractive Summarization (Optional)

For users seeking a more fluent, human-like summary, the application integrates an abstractive summarization model via the **Hugging Face** ‘transformers’ library.

- **Pre-trained Model:** The implementation uses the ‘t5-base’ model, a state-of-the-art sequence-to-sequence model well-suited for summarization tasks.
- **Robustness:** We use the high-level ‘pipeline’ from the library, which simplifies the process of tokenization, inference, and decoding. To handle very long transcripts that can exceed model limits, the input text is intelligently split into chunks, each chunk is summarized, and the resulting summaries are concatenated and summarized again for a final, coherent output.
- **Fallback System:** A key feature of our application is its resilience. If the abstractive model fails to load (due to memory constraints or missing dependencies), the system automatically falls back to the reliable extractive method, ensuring the user always receives a summary.

4 Conclusion and Future Scope

This project successfully demonstrates the power of NLP in solving real-world problems. The resulting application is a functional and robust tool for automatically summarizing YouTube video content, effectively saving users time and effort. By offering both extractive and abstractive methods, it caters to a variety of needs.

Future Scope While the current system is fully functional, there are several avenues for future enhancement:

- **Multi-language Support:** Extend the transcript fetching and summarization to support languages other than English.
- **Enhanced User Interface:** Improve the frontend to provide a more interactive and visually appealing experience.
- **Containerization:** Package the application using Docker to simplify deployment and ensure environmental consistency.