**Problem Statement:**

Build a text summarization tool that leverages a pre-trained large language model (LLM) to summarize long articles or documents.

Text summarization is a crucial task in enabling the distillation of large volumes of text into concise and informative summaries. This project focuses on developing a text summarization tool using Streamlit, a Python framework for building interactive web applications. The Large-Language-Model is main source to create the project and the use of the langchain.

**Methodology:**

***Streamlit:*** A Python framework that simplifies the creation of web applications with intuitive user interfaces.

*Transformers Library:* Specifically, the T5 model for conditional generation, along with its tokenizer for preprocessing text inputs.

*Langchain Library:* A custom library used for text preprocessing tasks, including document loading and segmentation.

**Modules:**

1. *langchain:*

The langchain module is used for the test processing task of splitting the text from the sentence or from the documents.

1. *transformers:*

The transformers module is a pre-trained model for various NLP tasks like text classification, text generation and summarization.

1. *T5Tokenizer:*

The T5Tokenizer is a component of the Hugging Face Transformers library specifically designed for tokenizing text inputs for models based on the T5 (Text-To-Text Transfer Transformer) architecture.

1. *torch:*

torch is a core Python library widely used for machine learning and deep learning tasks, particularly within the PyTorch framework.

1. *Base64:*

Base64 is a binary-to-text encoding scheme that represents binary data in an ASCII string format by converting it into a set of printable characters.

**Work Process:**

*1. Importing Libraries:*

The code begins by importing necessary libraries, including Streamlit (streamlit), the RecursiveCharacterTextSplitter from the langchain module, PyPDFLoader for loading PDF documents, and transformers for T5 model-related functionalities.

*2. Loading Model and Tokenizer:*

The code loads a pre-trained T5 model and tokenizer from a specified checkpoint directory using the T5Tokenizer.from\_pretrained() and T5ForConditionalGeneration.from\_pretrained() functions.

*3. File Preprocessing:*

The file\_preprocessing() function preprocesses the uploaded PDF file. It uses PyPDFLoader to load and split the PDF pages, and then RecursiveCharacterTextSplitter to split the text content of each page into smaller chunks.

*4. Text Summarization Pipeline:*

The llm\_pipeline() function creates a summarization pipeline using the pipeline() function from the transformers module. It specifies the T5 model, tokenizer, and parameters such as max\_length and min\_length for summarization.

*5. Displaying PDF:*

The displayPDF() function converts the uploaded PDF file into Base64 format and embeds it as an iframe in the Streamlit app using the markdown() function. This allows users to view the PDF content directly within the web app.

*6. Streamlit App Layout:*

The main() function sets up the layout of the Streamlit web app using st.title(), st.file\_uploader(), st.info(), st.markdown(), and st.success() functions.

It defines two columns (col1 and col2) to display the uploaded PDF file and the summarization result side by side.

*7. Handling User Input:*

The code checks if a PDF file has been uploaded by the user. If a file is uploaded, it reads the file, preprocesses it, displays the PDF in the first column, and shows the summarization result in the second column.

*8. Streamlit Configuration:*

The st.set\_page\_config() function configures the layout of the Streamlit app to be wide, allowing for a two-column display.

*9. Running the App:*

The code entry point executes the main() function, which sets up and runs the Streamlit web application.

**Implementation:**

The implementation of the text summarization tool encompasses the following key steps:

File Uploading: Users are provided with a file upload feature within the Streamlit interface, enabling them to upload PDF documents for summarization.

Document Preprocessing: Upon uploading a PDF document, the Langchain library is utilized to preprocess the text. The document is segmented into smaller chunks to facilitate efficient summarization.

Summarization Pipeline: The preprocessed text chunks are fed into a T5 model fine-tuned for summarization using the Transformers library. The model generates abstractive summaries of the input text, capturing its essential meaning.

Displaying Results: The generated summaries are presented to the user via the Streamlit interface in real-time, allowing for easy access and evaluation.