# AI-POWERED TEXT SUMMARIZER

## OVERVIEW

The AI Text Summarizer is a Python application that generates concise summaries from large text files and PDFs.
It supports **Abstractive Summarization** using the **T5-small transformer model**, producing human-like summaries by rephrasing content.
**Extractive Summarization** uses the **LexRank algorithm** to select the most important sentences directly from the text.
The tool can process **multiple TXT or PDF files at once**, making batch summarization efficient.
Long documents are automatically **split into chunks** to ensure smooth and accurate summarization.
Users interact via a **Streamlit web interface** for uploading files or entering text manually.
The project demonstrates **modern NLP techniques, transformer-based models, and classic extractive methods**.
It provides a practical, user-friendly solution for quickly understanding large volumes of text.

## SETUP INSTRUCTIONS

**1.Clone the Repository**

Open a terminal or command prompt.Run the following commands:

```
git clone <your-repo-url>
```

```
cd ai_text_summarizer
```

**2. Create and Activate a Virtual Environment**

*Windows:*

```
python -m venv .venv
```

```
.venv\Scripts\activate
```

*macOS / Linux:*

```
python -m venv .venv
```

```
source .venv/bin/activate
```

**3. Install Required Libraries**

```
pip install -r requirements.txt
```

4. Download NLTK Tokenizer Data

```
python -c "import nltk; nltk.download('punkt')"
```

**5. Run the Streamlit Web App**

```
streamlit run streamlit_app.py
```

- Open your browser at http://localhost:8501
- Upload text or PDF(s) and choose summarization settings

# Libraries Required:

 **transformers** – For abstractive summarization using BART or T5

 **torch** – Backend for Hugging Face models

 **sumy** – For extractive summarization (LexRank)

 **nltk** – Tokenizer for extractive summarization

 **streamlit** – Optional web interface

 **PyPDF2** – To read PDF files for summarization

 **rouge-score** – For evaluation of summaries

# Usage Guide for AI Text Summarizer

**1. Command Line Interface (CLI)**

**Single text or file**

Summarize a .txt file using abstractive summarization (BART or T5):

```
python summarizer.py --input "path/to/article.txt" --model_name
facebook/bart-large-cnn --min_length 60 --max_length 180
```

Or pass raw text directly:

```
python summarizer.py --text "Paste your long text here..." --model_name t5-
small --max_length 130
```

**Extractive mode**

Summarize using LexRank (extractive):

```
python summarizer.py --input "article.txt" --mode extractive --sentences 4
```

**Batch folder of .txt files**

Summarize multiple files in a folder:

```
python summarizer.py --batch-dir "data/input_texts" --mode abstractive --
model_name facebook/bart-large-cnn --max_length 180
```

This generates one .summary.txt file per input file in the folder.

**Compare models**

Compare BART, T5, and LexRank summaries side-by-side:

```
python summarizer.py --input "article.txt" --compare
```

**2. Streamlit Web App**

Start the app:

```
streamlit run streamlit_app.py
```

# Models Used & NLP Approach

**1. Abstractive Summarization (Transformer-based)**

- **Models:**
  - **Facebook BART-large-CNN:** A sequence-to-sequence transformer pre-trained for summarization tasks. Good for generating coherent and concise summaries.
  - **T5-small:** A lightweight transformer model suitable for faster summarization with smaller resource requirements.
- **Approach:**
  - Long text is split into chunks to fit the model's input size.
  - Each chunk is summarized individually, and partial summaries are combined.
  - Optional beam search (num_beams) helps generate higher-quality summaries by exploring multiple possible outputs.

**2. Extractive Summarization (LexRank)**

- **Model:** LexRank (from the sumy library), an unsupervised graph-based algorithm.
- **Approach:**
    - Sentences are represented as nodes in a similarity graph.
    - Central sentences with highest importance scores are selected as the summary.
    - No model training or internet connection is required.

**3. NLP Techniques Used**

- **Tokenization:** Using nltk for sentence splitting (needed for LexRank).
- **Chunking:** For transformer models to handle long texts.
- **Pipeline abstraction:** Hugging Face's pipeline("summarization") manages tokenization, model inference, and decoding.
- **Evaluation :** ROUGE metrics to compare generated summaries with reference summaries.

# Features

1. **Abstractive Summarization**
    - Uses transformer models (BART-large-CNN or T5-small) to generate concise, human-like summaries.
2. **Extractive Summarization**
    - Uses LexRank to select the most important sentences from the text without generating new words.
3. **Supports Multiple Input Types**
    - Raw text input, .txt files, and .pdf documents.
4. **Batch Processing**
    - Summarize multiple .txt files at once, producing individual summaries for each file.
5. **Streamlit Web Interface**
    - User-friendly interface to paste text, upload files, and view summaries.
6. **Customizable Parameters**
    - Min/max length for abstractive summaries, number of sentences for extractive summaries, and beam search options.
7. **Performance Evaluation**
    - Supports ROUGE metrics (ROUGE-1, ROUGE-2, ROUGE-L) to compare generated summaries with reference summaries.
8. **Handles Long Documents**
    - Automatically chunks long texts to fit transformer model context windows.
9. **Lightweight & Flexible**
    - Can run on CPU, with optional GPU support for faster summarization.

# Future Improvements

- **Support for more models:** Integrate larger transformer models (e.g., BART-large, Pegasus) for higher-quality abstractive summaries.
- **Multi-language support:** Extend summarization to texts in languages other than English.
- **Advanced extractive methods:** Incorporate other graph-based or embedding-based extractive techniques for better sentence selection.
- **Customizable summary length:** Allow users to define summary length dynamically based on their needs.
- **Improved UI/UX:** Enhance Streamlit interface with themes, live preview, and batch progress bars.
- **Offline model caching:** Enable offline summarization to reduce dependency on internet connectivity.
- **Integration with document formats:** Support DOCX, PPTX, and web pages as input sources.
- **Evaluation dashboard:** Include ROUGE, BLEU, and readability metrics in the web interface.

# Conclusion

The **AI Text Summarizer** efficiently condenses large texts using modern NLP techniques. It combines **abstractive summarization** (T5-small / BART) with **extractive summarization** (LexRank) for flexible output. The tool supports multiple input types, batch processing, and an optional web interface, providing a practical, user-friendly solution for generating accurate summaries.