The `FileLingo` Streamlit application is a powerful tool that allows users to upload PDF or image files, extract text from them using OCR, summarize the extracted text, and translate the summary into a different language. The application leverages several libraries and pre-trained models for OCR, text processing, summarization, and translation.

This report details how to set up and run the `FileLingo` application on your local machine.

1. Environment Setup

Before running the application, you need to set up your environment with the necessary libraries and tools.

1.1 Install Python

Ensure you have Python installed on your system. Python 3.8 or higher is recommended.

1.2 Install Required Libraries

You can install the required Python libraries by running the following command in your terminal:

!pip install streamlit pdf2image pymupdf unidecode torch transformers pypdf2 pytesseract pillow nltk pyspellchecker

- `streamlit`: For creating the web application interface.
- `pdf2image`: To convert PDF files to images (if needed).
- `fitz` (PyMuPDF): To extract text from PDFs.
- `unidecode`: For text normalization.
- `torch` and `transformers`: For loading and running the summarization and translation models.
- `PyPDF2`: To handle PDF files.
- `pytesseract`: For Optical Character Recognition (OCR) on images.
- `PIL` (Pillow): For handling image files.
- `re`, `nltk`, `spellchecker`: For text cleaning and processing.

1.3 Download Tesseract

Tesseract is an OCR engine used by `pytesseract`. You need to install Tesseract on your system:

- **Windows:** Download and install from Tesseract at UB Mannheim.
- **macOS:** Install using Homebrew: `brew install tesseract`.
- **Linux:** Install using apt: `sudo apt-get install tesseract-ocr`.

1.4 Download NLTK Data

To use the NLTK library, you need to download the required data:

```
import nltk
nltk.download('punkt')
```

2. Running the Application

Once your environment is set up, you can run the application using the following steps:

2.1 Create the Application File

Save the provided code in a Python file named `app.py`:

```
%%writefile app.py
# Paste the provided code here
```

2.2 Run the Streamlit Application

Use the terminal to navigate to the directory containing `app.py` and run the application:

```
streamlit run app.py
```

2.3 Using the Application

1. **Upload File**: Use the "Upload PDF or Image" button to upload a PDF or an image file.
2. **Select Languages**: Choose the source language (language of the uploaded file's content) and the target language (language you want the summary to be translated into) from the dropdown menus.
3. **Run the Process**: Click the "Run" button to start the extraction, summarization, and translation process.
4. **View the Results**: The translated summary will be displayed under "Translated Summary".

3. Code Overview

- **Text Extraction**: Depending on the file type, the text is extracted using either `PyMuPDF` for PDFs or `pytesseract` for images.
- **Text Cleaning**: The extracted text is cleaned using custom functions `clean_text1` and `clean_text2` to improve the quality of the input for summarization.
- **Summarization**: The cleaned text is summarized using a pre-trained BART model.

- **Translation**: The summary is split into smaller chunks and translated into the target language using the mBART model.
- **Display**: The translated summary is displayed on the Streamlit app interface.

4. Additional Features

- **Caching**: The functions for extracting text from PDFs and images are cached using `@st.cache_data` to improve performance.
- **Language Support**: The application supports translations between many languages, thanks to the mBART model's multilingual capabilities.
- **Spell Check and Correction**: The application includes a spell checker to enhance the accuracy of the text processing.

5. Limitations and Notes

- **Performance**: Processing large documents or high-resolution images might be slow depending on your system's hardware.
- **Model Directory**: Ensure that the model directory for the summarization model is correctly specified and that the model files are accessible.
- **Tesseract Configuration**: Make sure that the `pytesseract` is correctly configured to use the installed Tesseract engine.

This setup should allow you to successfully run and use the `FileLingo` application for OCR, summarization, and translation tasks.

Current Deployment Status

Important Note: The deployment of the `FileLingo` application is still in progress. I am currently facing limitations due to insufficient cloud storage for hosting the app. This means that while the application runs locally, I am working on resolving these storage issues to enable cloud deployment.