How to install & RUN

Step by step process:

1. Install Python 3.11.5 x64bit windows from `https://www.python.org/downloads/release/python-3115/` make sure install with Path. During installation, make sure to check the box that says "Add Python to PATH" before clicking the install button
2. Install VS Code `https://code.visualstudio.com/Download` you can also add Extebnsions for viweing Path Intellisense, isort, Pylance and python syntax from extension marketplace in VS Studio code to make experience better
3. Run this command `pip install -r requirements.txt` it will install dependencies
4. Optional step: Delete Existing Database and uploaded files from upload folder if you want train from own dataset. after this yu will need upload at least 30 PDFs for training properly
5. Run the APP `python app.py`
6. Open it in port 8888 `http://127.0.0.1:8888/`
7. Upload the PDF Files

Explanation

Overview of the Flask Application

The Flask application you've provided is designed to upload PDF documents, extract text from them, summarize the text using a pre-trained model, cluster the documents using Bisecting K-means clustering, and evaluate the performance of a Multinomial Naive Bayes classifier on the clustered data.

File Structure

- **app.py**: This is the main Flask application file. It initializes the Flask app, sets up routes, handles file uploads, text extraction, summarization, clustering, and classification.
- **index.html**: The home page template where users can upload PDF files.
- **summary.html**: Template to display the summary of a document after uploading.
- **clusters.html**: Template to display clusters and associated documents.
- **error.html**: Template to display error messages if text extraction or summarization fails.

Workflow Explanation

1. **Initialization (`app.py`)**:
   - **Flask Setup**: Initializes the Flask app and configures settings like upload folder (`UPLOAD_FOLDER`).
   - **Database Initialization (`init_db()`)**: Sets up SQLite database (`database.db`) to store document metadata, including filename, size, upload date, extracted text, summary, and cluster.
2. **Text Extraction (`extract_text_from_pdf()`)**:
   - Uses PyMuPDF (`fitz`) to extract text from uploaded PDF files.

3. **Text Summarization (`upload_file()` route)**:
   - Summarizes the extracted text using Hugging Face's transformers library (`pipeline('summarization')`).

4. **Data Storage**:
   - Stores document metadata and extracted information (filename, size, upload date, text, summary, cluster) in SQLite database (`insert_document()`).

5. **Clustering (`upload_file()` route)**:
   - **Text Vectorization**: Uses `TfidfVectorizer` to convert text data into TF-IDF features.
   - **Optimal Cluster Determination**: Determines the optimal number of clusters (`k`) using silhouette score (`determine_optimal_clusters()`).
   - **Bisecting K-means Clustering**: Uses `BisectingKMeans` to cluster documents into `k` clusters.

6. **Classification (`upload_file()` route)**:
   - **Train-Test Split**: Splits the data into training and testing sets using `train_test_split()`.
   - **Classifier Training**: Trains a Multinomial Naive Bayes classifier (`MultinomialNB`) on the training set.
   - **Prediction and Evaluation**: Predicts cluster labels on the test set, evaluates performance using precision and recall scores (`precision_score()`, `recall_score()`).

7. **Logging**:
   - Logs various events, errors, and debug messages to `app.log` for debugging and monitoring (`logging`).

8. **Web Interface**:
   - **Home Page (`/`)**: Allows users to upload PDF files.
   - **Summary Page (`/upload`)**: Displays document summary and clustering/classification results.
   - **Cluster Page (`/clusters`)**: Displays clusters and associated documents.

Notes:

- **Error Handling**: The application includes basic error handling for file uploads, text extraction, summarization, clustering, and classification.

- **Database Usage**: SQLite is used for storing document metadata. Each document's cluster assignment is updated after clustering.

- **Performance**: The application provides basic performance metrics (precision and recall) for the clustering and classification tasks.

This setup allows for a streamlined process of uploading, processing, summarizing, clustering, and evaluating PDF documents within a web application using Flask and various machine learning and natural language processing libraries.