Persona-Driven Document Intelligence

This project is part of **Challenge Round 1B** and focuses on analyzing multiple UPI fraud detection research documents using AI. The goal is to **automatically identify the most relevant sections** of provided research PDFs that help answer a specific job/task — such as determining effective machine learning algorithms for UPI fraud detection.

Project Structure

```
.
├── app/
│   ├── input/                   # Input PDFs to be analyzed
│   ├── output/                  # Output folder containing JSON results
│   ├── utils/                   # Utility modules for PDF parsing, embedding, ranking, etc.
│   │   ├── embedder.py          # SentenceTransformer wrapper
│   │   ├── formatter.py         # JSON result builder
│   │   ├── pdf_parser.py        # Extracts text from PDF pages
│   │   └── ranker.py            # Ranks sections based on relevance
│   ├── main.py                  # Main script to run the analysis
├── test_input/
│   └── challenge1b_input.json   # Contains metadata, persona, job, and document list
├── output/
│   └── challenge1b_output.json  # Final generated output
├── approach_explanation.md      # (Optional) Describes approach taken (not executed)
├── Dockerfile                   # Docker setup for containerized execution
├── requirements.txt             # Python dependencies
```

Problem Statement

> **Task**: Build an intelligent document analysis system that extracts and ranks the top 5 most relevant sections from a set of academic PDFs, customized for a specific persona and their job-to-be-done.

Use Case

- **Persona**: Fraud Analyst at a FinTech company
- **Job-To-Be-Done**: Identify the most effective **machine learning algorithms and techniques** for detecting **UPI (Unified Payments Interface) fraud**.

Key Requirements

- The system should analyze a **collection of academic PDFs**.
- It must extract content at the **section level** (not just paragraphs or keywords).
- The top **5 most relevant sections** should be selected and ranked based on their alignment with the persona's job.
- The analysis must be **persona-aware**, meaning it should account for the background, goals, and needs of the specific role (Fraud Analyst).
- The relevance should be evaluated based on **semantic context**, not just keyword matching.

How It Works

The pipeline performs the following steps:

1. Input Loading
- Reads metadata from `challenge1b_input.json`, which includes:
  - List of documents (PDFs)
  - Persona (user role)
  - Job/task description

2. PDF Parsing (`utils/pdf_parser.py`)
- Uses PyMuPDF to extract text from each page of each PDF.
- Stores each page as a section with:
  - Page number
  - Text content
  - Title: `Page X`

3. Embedding (`utils/embedder.py`)
- Uses `sentence-transformers` model (`all-MiniLM-L6-v2`) to convert texts into embeddings.
- Also embeds the job/task description.

4. Ranking (`utils/ranker.py`)
- Computes cosine similarity between the query embedding and section embeddings.
- Selects top 5 sections with the highest similarity.

5. Formatting Output (`utils/formatter.py`)
- Builds a JSON output with:
  - Metadata
  - Ranked extracted sections (titles, page numbers)
  - Subsection analysis with top 2000 characters from each section

6. Output Saving
- Result is saved to `app/output/challenge1b_output.json`.

Dockerized Setup

Run the pipeline using Docker:

Build the Image:
```bash
docker build -t upi-fraud-analyzer .
```

Run the Container:
```bash
docker run --rm -v $(pwd)/app/output:/app/app/output upi-fraud-analyzer
```

This mounts the output folder so results persist outside the container.

Requirements

Install dependencies locally (if not using Docker):

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

Key dependencies:
- `sentence-transformers`
- `torch`
- `PyMuPDF`

Input Example (`test_input/challenge1b_input.json`)

```json
{
  "persona": {
    "role": "Fraud Analyst at a FinTech company"
  },
  "job_to_be_done": {
    "task": "Identify the most effective machine learning algorithms, preprocessing methods,
and evaluation techniques to build a real-time UPI fraud detection system."
  },
  "documents": [
    {
      "filename":
"Comparative_Analysis_of_UPI_Fraud_Detection_Using_Ensemble_Learning.pdf",
      "title": "Comparative Analysis of UPI Fraud Detection Using Ensemble Learning"
    },
    {
      "filename": "Fraud_Fighters_-_How_AI_and_ML_are_Revolutionizing_UPI_Security.pdf",
      "title": "Fraud Fighters - How AI and ML are Revolutionizing UPI Security"
    },
    {
      "filename": "Secure_UPI_Machine_Learning-
Driven_Fraud_Detection_System_for_UPI_Transactions.pdf",
      "title": "Secure UPI - ML Driven Fraud Detection System"
    }
  ]
}
```

Output Format (`output/challenge1b_output.json`)

```json
{
  "metadata": {
    "input_documents": [...],
    "persona": "...",
    "job_to_be_done": "...",
    "processing_timestamp": "YYYY-MM-DDTHH:MM:SS"
  },
  "extracted_sections": [
    {
      "document": "...",
      "section_title": "...",
      "importance_rank": 1,
      "page_number": 3
    }
  ],
  "subsection_analysis": [
    {
      "document": "...",
      "refined_text": "Top 2000 chars...",
      "page_number": 3
    }
  ]
}
```

✅ Summary

- Multi-document processor for UPI fraud literature
- Uses semantic embeddings to rank sections by relevance
- Persona-aware, job-guided information retrieval
- Fully Dockerized, portable, and reproducible
