Solution Architecture Documentation

1. Overview

The Python script implements an AI-powered document and email processing solution using Streamlit for UI interaction. It leverages machine learning and NLP techniques to extract, classify, and analyze textual content from uploaded files.

2. Key Components & Interactions

2.1 User Interface (UI)

- Built using Streamlit.
- Users can upload multiple files (PDF, DOCX, and EML) for processing.
- Sidebar allows users to configure extracted fields.
- Displays processed results and duplicate detection feedback.
- Implements custom CSS for enhanced styling.

2.2 File Processing Module

Handles file uploads and extraction of textual content from different formats:
- PDF Processing: Uses pdfplumber to extract text.
- DOCX Processing: Uses python-docx to extract text.
- Email Processing: Uses email.parser to extract email body and attachments.
- HTML Parsing: Uses BeautifulSoup to extract text from HTML email bodies.

2.3 Hugging Face API Integration

- Uses Hugging Face Transformers API to classify email intent and extract structured data.
- Calls API endpoints with a prompt-based approach to extract relevant fields.
- Parses API responses and converts extracted data into structured JSON format.

2.4 Text Preprocessing & Similarity Detection

- Uses scikit-learns TfidfVectorizer to convert text into vector representations.
- Applies cosine similarity to detect duplicate files with a threshold of 0.85.
- Utilizes NLTK stop words to preprocess and clean text.

Solution Architecture Documentation

3. Design Patterns Used

3.1 Factory Pattern (for File Processing)

- Different file types are handled by specialized extraction functions.
- The program dynamically calls the appropriate function based on file extension.

3.2 Adapter Pattern (for API Integration)

- Hugging Face API responses are parsed and converted into JSON.
- Ensures consistent data format across different API responses.

3.3 Pipeline Pattern (for NLP Processing)

- Preprocessing Classification Extraction Similarity Detection Display Results.
- Each stage processes the output of the previous step and feeds it forward.

4. Libraries & Frameworks Used

Library	Purpose
Streamlit	UI development
pdfplumber	PDF text extraction
docx	DOCX text extraction
email.parser	Email body & attachment extraction
BeautifulSoup	HTML parsing for emails
transformers	Hugging Face NLP model access
scikit-learn	Text vectorization & similarity detection
pandas	Data processing & display

5. Conclusion

This script integrates AI-driven classification and data extraction within a user-friendly UI. By leveraging NLP, machine learning, and API-driven orchestration, it efficiently processes documents and emails for structured data extraction and triage.