Build a Resume Screening Application with LangChain

Overview

This Resume Screening Application is built using LangChain to help HR professionals efficiently evaluate candidate resumes against job requirements. The application leverages LangChain components and LangChain Expression Language (LCEL) to create an automated, AI-powered resume analyzer. It compares uploaded resumes with specified job descriptions, generates structured insights, assigns a suitability score, and stores the results using vector embeddings with ChromaDB for future retrieval.

Problem Statement

The objective is to build a Resume Screening Application using LangChain that allows HR managers to input job requirements and submit candidate resumes in PDF, DOCX, or TXT formats. The system extracts text from the resumes, compares it against job requirements using an AI model, and produces a structured analysis. The analysis includes a match score, skills assessment, experience relevance, education evaluation, strengths, weaknesses, and an overall candidate recommendation. The application should be user-friendly, support multiple file formats, and allow users to download the analysis report for record-keeping.

Approach for the Solution

The Resume Screening Application uses multiple LangChain components to deliver an end-to-end automated screening solution. The approach focuses on streamlining resume evaluation while maintaining flexibility and scalability.

Data Ingestion: HR managers can upload resumes in PDF, DOCX, or TXT formats. Text is extracted using appropriate loaders such as PyPDFLoader and Docx2txtLoader.
Text Processing: Extracted text is split into manageable chunks using RecursiveCharacterTextSplitter to fit model token limits.
Model Inference: A generative AI model (ChatGoogleGenerativeAI) evaluates how well the resume matches job requirements and produces a structured analysis, including a suitability score.
Embeddings and Semantic Search: The generated analysis is converted into vector embeddings using GoogleGenerativeAIEmbeddings and stored in ChromaDB to enable semantic search and historical retrieval.
User Interface: The application uses Streamlit to provide an interactive UI where HR professionals can input job requirements, upload resumes, view AI-generated insights, and download reports.
Pipeline Construction with LCEL: The complete workflow is modularized using LangChain Expression Language (LCEL), ensuring a smooth, maintainable, and flexible processing pipeline.

Structure of the Solution

The solution architecture is modular and clearly separates responsibilities across LangChain components:
- Resume Upload & Job Requirements: Collected via the Streamlit interface.
- Text Extraction: Performed using file-type-specific loaders.
- Text Processing: Managed by RecursiveCharacterTextSplitter.
- Model Inference & Analysis: Handled by the LLM (ChatGoogleGenerativeAI).
- Embedding and Storage: Analysis is embedded and stored in ChromaDB.
- User Interface: Streamlit presents results and supports report downloads.