Project-4:- Resume Matching & Shortlisting System

Overview

Developing a multi-agentic AI system that can automatically read and summarize job descriptions (JDs), match candidate qualifications with the JDs, shortlist candidates and store shortlisted candidate details in a database.

Important Steps

→ Extracting text and analyzing details from multiple resumes in PDF format.

→ Calculating similarity scores of uploaded resumes with the job description.

→ Based on the similarity scores, top-5 resumes are shortlisted.

→ Shortlisted candidate details are stored in a database.

Dataset

→ Resumes dataset containing 200 resumes of different candidates with different backgrounds.

→ A CSV file containing different job titles and their descriptions.

Llama 3.2:1b

→ Llama stands for Large Language Model Meta AI
It's a series of AI language models developed by Meta, similar to OpenAI's GPT models.

→ Llama 3.2:1b is the 2nd version of the 3rd generation of the Llama models, containing 1 Billion parameters.

NOTE:- Parameters are like "neurons" of the model i.e., the more parameters, the smarter and more capable the model generally is.

Features of Llama3.2:1b

i. Lightweight Tasks:- Fast response, low memory use.

ii. Edge Devices:- Phones, laptops, maybe smaller devices, not huge server farms

iii. Fine-tuning easily:- It can be trained on small, specific datasets for tasks like custom chatbots or assistants.

iv. Lower Cost:- 1 billion parameters means it's cheaper to run compared to giant models like GPT-4.

Advantages

→ Better understanding of complex language

→ Improved instruction following

→ More efficient tokenization

→ Trained with large, high-quality datasets

→ Available with an open license

Model Building (In-Detail)

I Importing Libraries

→ PyMuPDF (fitz)

→ SpaCy

→ Regular Expression (re)

→ Ollama

→ mysql.connector

→ Streamlit

→ subprocess

II Loading Models

→ Loads a small english model from SpaCy for text processing.

→ Sets the name of the LLM model for similarity calculation.

III Text Extraction and Preprocessing

→ Reads PDF file page-by-page.

→ Extracts all plain text from each page

→ Converts text to lowercase

→ Removes all non-word characters

→ Uses SpaCy to

  → Lemmatize words (removing suffix/prefix)

  → Removing stopwords (the, is, and, etc)

→ Returns clean, simplified text

IV Calculating similarity using LLM

→ Sends JD and preprocessed text to Llama model

→ Asks the model to give similarity score (1-100)

→ Extracts the first number it finds the model's response.

→ Returns the score on a 0-1 scale (ex: 80 → 0.8)

V Setup Database Connection

→ Connects to a MySQL database.

→ For each top resume, it inserts:

- Job Description
- Resume Name
- Matching Score

→ Saves this data into a table.

VI Streamlit App UI

a. Web building

→ Displays app title and instructions

→ Text area for user to paste JD

→ File uploader for uploading multiple PDF resumes.

b. Resume Shortlisting Logic

→ When the "Match Resumes" button clicked.

- Check if JD and resumes are uploaded
- Preprocess JD text
- Loop through each resume:
  - → Extract text
  - → Preprocess text
  - → Calculate similarity score
  - → Store name & score

→ Sort resumes by scores in ascending order

→ Display top 5 resumes with scores

→ Store them in database.