Crux Internship Assignment

Md Aakib Alam Ansari

February 23, 2024

1. OVERVIEW:

The CV Shortlister System is a comprehensive solution designed to automate the process of shortlisting resumes based on job descriptions. This report provides insights into the approach, implementation details, integration of a React-based frontend, use of AJAX polling for progress tracking, concurrency for efficiency and deployment on Azure App Services.

2. IMPLEMENTATION DETAILS:

- **Framework:** Built using Django, a high-level Python web framework, for rapid development and clean, maintainable code.
- **Modules:** The application is structured into multiple modules:
  - `api`: Handles API endpoints and logic for parsing resumes and extracting information.
  - `views.py`: I defined the route handler function here responsible for processing incoming requests from the frontend and used concurrency for extracting each section concurrently to reduce expected time per resume processing.
  - `projects.py, education.py, workex.py`: Modules responsible for extracting projects, education, and work experience sections from resumes, respectively, using Google's generative AI.
  - `embeddings.py`: This module calculates similarity scores between textual data using pre-trained models from Google's generative AI library. It generates embeddings for input text and computes similarity scores using cosine similarity.
  - `intro.py`: Contains functions to extract basic information like name and email from resumes using regular expressions.
- **Dependencies:** Utilizes PyPDF2 for parsing PDF resumes, NLTK for text processing tasks, and Google's generative AI library for content generation and extraction.
- **API Endpoint:** Provides a single endpoint '/submit' for receiving job descriptions and resumes, processing them, and returning shortlisted candidates with relevant information. Additionally, the system utilizes the '/poll' endpoint for receiving real-time progress updates during resume processing.

3. HYPERPARAMETERS:

- **Generative AI Configuration:** Hyperparameters such as temperature, top_p, top_k, and max_output_tokens are configured for Google's generative AI model. These parameters control the generation of content and the length of the output text.

4. INTEGRATION WITH REACT FRONTEND:

- **Frontend Development:** A React-based frontend application has been developed to interact with the backend server. This frontend provides a user-friendly interface for users to input job descriptions and upload resumes for shortlisting.
- **Github repository:** https://github.com/AakibAlam/cv-shortlister-frontend
- **Deployment:** The project has been deployed on Azure App Services for testing and accessibility purposes. https://parse.cvninja.studio/

5. DEPLOYMENT ON AZURE:

- **Successful Deployment:** The project has been successfully deployed on Azure App Services for hosting the backend server.
- **Integration:** The frontend and backend components are deployed independently and harmoniously integrated to facilitate cohesive interaction and seamless operation.

6. EVALUATION RESULTS:

- **Relevancy Score Calculation:** To determine the relevancy score for each resume, the system identifies the top three projects and work experiences extracted from the resume. These top three projects and work experiences are then combined, and a combined embedding is generated. Subsequently, the system computes the similarity between this combined embedding and the provided job description to derive the final relevancy score.

7. INSIGHTS GAINED:

- **Deployment Challenges:** Deploying complex applications involving frontend and backend components requires thorough testing and debugging to ensure seamless integration.
- **Cloud Deployment:** Deploying on cloud platforms like Azure App Services offers scalability and accessibility benefits but requires careful configuration to ensure compatibility between frontend and backend components.
- **Effective Utilization of AI:**Leveraging Google's generative AI model enables automated extraction of detailed information from resumes, enhancing the efficiency of the shortlisting process.
- **Challenges:** Handling variations in resume formats and structures poses challenges in accurate parsing and extraction. Regular updates and refinements to the parsing algorithms are necessary to handle diverse resume styles.
- **Scalability:** The modular structure of the application allows for easy scalability and maintenance. Additional features and improvements can be integrated seamlessly to enhance functionality and performance.

8. CONCLUSION:

The CV Shortlister Backend, coupled with its React-based frontend, presents an effective solution for automating resume shortlisting. With successful deployment and seamless integration between the backend and frontend on Azure App Services, recruiters now have

a robust tool at their disposal. The system allows users to effortlessly upload resumes and job descriptions, enabling the identification of top candidates based on relevance to the job requirements.

Link to deployment: https://parse.cvninja.studio/