Infosys Internship 4.0 Project Documentation

Title: Project Documentation: AI Resume Matchmaker

•Introduction:

In response to the challenges posed by traditional resume screening methods, the AI Resume Matchmaker project introduces an innovative solution leveraging cutting-edge artificial intelligence technologies. This project aims to transform the recruitment process by automating the extraction, analysis, and alignment of critical information from resumes and job descriptions.

The primary goal is to develop a robust AI model capable of precisely identifying and matching key qualifications, skills, and experiences required for specific job roles. By harnessing natural language processing (NLP) and machine learning techniques, the AI Resume Matchmaker enhances the efficiency and accuracy of candidate evaluation. This technological advancement not only accelerates the hiring process but also ensures a more objective and standardized approach to candidate selection.

Project Objectives

The key objectives of the AI Resume Matchmaker project include:

- **Advanced Information Extraction**: Implement advanced NLP techniques to extract and categorize relevant information from resumes and job descriptions.
- **Intelligent Matching**: Develop machine learning algorithms that intelligently match candidate profiles with job requirements based on extracted attributes.
- **Scalability and Performance**: Design scalable architecture and optimize performance to handle large volumes of resumes and job descriptions efficiently.

Project Scope:

The AI Resume Matchmaker project aims to automate and enhance the process of matching job descriptions with candidate resumes using artificial intelligence technologies. The project encompasses several key components and considerations:

Inclusions:

1. **Resume and Job Description Processing:**
    - Extraction of textual content from PDF resumes and job descriptions.
    - Text preprocessing including cleaning, normalization, and vectorization.
2. **AI Model Development:**
    - Utilization of Sentence Transformers for encoding resume and job description texts.
    - Implementation of cosine similarity to determine the match score between resumes and job descriptions.
3. **Qdrant Integration:**
    - Establishment of a Qdrant collection for storing and querying vector representations of resumes and job descriptions.
    - Integration of Qdrant for efficient storage and retrieval of vector data.

Exclusions:

1. **Real-Time Data Integration:**
    - The project does not currently include real-time data integration for live job postings or dynamically changing resume databases.
2. **Advanced Natural Language Understanding:**
    - While the project utilizes basic NLP techniques for text preprocessing and vectorization, it does not incorporate advanced semantic understanding or context-aware processing.
3. **Extensive Backend Development:**
    - Backend infrastructure for scaling beyond the current prototype phase is considered out of scope for this iteration.

Limitations and Constraints:

1. **Computational Resources:**
    - Limited computational resources may constrain the size and complexity of datasets that can be processed within reasonable time frames.
2. **Accuracy and Generalization:**
    - The accuracy of resume-job description matching heavily depends on the quality and completeness of the textual information extracted and encoded from resumes and job descriptions.
3. **Deployment Environment:**
    - The deployment and operational environment may impose constraints on the integration and performance of the AI model and Qdrant database.

Considerations:

1. **Data Privacy and Security:**
    - Measures are taken to ensure the privacy and security of uploaded resumes and job descriptions during processing and storage.
2. **User Interface and Experience:**
    - The project aims to provide a streamlined user interface through Streamlit for uploading and processing documents, with minimal user intervention required.

By defining these boundaries and considerations, the project aims to deliver a focused solution that enhances recruitment processes while acknowledging its current limitations and constraints.

•Requirements:

Functional Requirements:

1. **Document Upload and Processing:**
    - Users can upload PDF documents containing resumes and job descriptions.
    - The system extracts text from uploaded PDFs and preprocesses it for further analysis.
2. **Text Preprocessing:**
    - Text cleaning: Removal of punctuation, URLs, emails, and non-alphabetic characters.
    - Text normalization: Lowercasing, lemmatization, and removal of stop words.
3. **AI Model Integration:**
    - Utilization of Sentence Transformers for encoding textual information into numerical vectors.
    - Implementation of cosine similarity to calculate the matching score between encoded vectors of resumes and job descriptions.
4. **Qdrant Database Integration:**
    - Establishment of a Qdrant collection for storing and querying vector representations of resumes and job descriptions.
    - Integration of Qdrant API for efficient storage and retrieval operations.
5. **User Interface (UI):**
    - Development of a Streamlit-based web application for seamless document upload, processing, and display of matching scores.
    - Display of top matching job descriptions and their respective scores.

Non-Functional Requirements:

1. **Performance:**
    - Efficient processing of documents within a reasonable time frame, even with large PDF files.
    - Low latency in calculating and displaying matching scores using the AI model.
2. **Scalability:**
    - The system should handle an increasing number of users and document uploads without significant degradation in performance.

3. **Security:**

- Ensuring data privacy and confidentiality of uploaded resumes and job descriptions during processing and storage.
- Implementing secure communication channels and data encryption where applicable.

3. **Usability:**

- Intuitive user interface design to facilitate easy document upload and result visualization.
- Minimal user interaction required beyond initial document upload.

User Stories

User Story 1: Simplified Candidate Screening

As a Recruiter, I want to efficiently screen through numerous resumes to identify top candidates, so I can focus my time on interviewing the best fits for our open positions.

Acceptance Criteria:

- Upload multiple resumes and job descriptions to the AI Resume Matchmaker system.
- Receive a clear and concise list of top-matching candidates based on skills and qualifications.
- Save time and effort previously spent on manual screening.

User Story 2: Enhanced Hiring Decision Making

As a Hiring Manager, I need reliable insights into how well candidates' skills align with our job requirements, so I can make informed hiring decisions swiftly.

Acceptance Criteria:

- Access a dashboard displaying detailed similarity scores between resumes and job descriptions.
- Utilize data-driven insights to prioritize candidates who closely match our specific job criteria.
- Improve the accuracy and efficiency of our hiring process.

Use Cases

Use Case 1: Uploading and Processing Documents

Description: The AI Resume Matchmaker system processes uploaded resumes and job descriptions to extract, clean, and normalize text for accurate matching.

Actors: Recruiter, AI Resume Matchmaker System

Flow:

1. **Upload Documents:**
    - **Actor Action:** The Recruiter uploads multiple PDF documents containing resumes and job descriptions.
    - **System Response:** The AI Resume Matchmaker system receives and verifies the document format.
2. **Text Processing:**
    - **Actor Action:** The system extracts text from each document and cleans it by removing punctuation, URLs, and non-alphabetic characters.
    - **System Response:** Cleaned text is normalized to ensure consistency across all documents.
3. **Encoding and Vectorization:**
    - **Actor Action:** The system utilizes advanced AI models like Sentence Transformers to encode text into numerical vectors.
    - **System Response:** Vectors represent semantic meanings and are used to calculate similarity scores between resumes and job descriptions.

Use Case 2: Matching and Scoring

Description: The AI Resume Matchmaker system calculates similarity scores to identify the best-fit candidates for specific job roles.

Actors: Recruiter, AI Resume Matchmaker System

Flow:

1. **Cosine Similarity Calculation:**
    - **Actor Action:** The system computes cosine similarity scores between vector representations of resumes and job descriptions.
    - **System Response:** Scores indicate the degree of alignment between candidate skills and job requirements.
2. **Display Results:**
    - **Actor Action:** The recruiter views a ranked list of top-matching candidates and their respective similarity scores.
    - **System Response:** Results are presented on a user-friendly interface, allowing recruiters to make informed decisions efficiently.

•Technical Stack:

Programming Languages:

- Python

Frameworks/Libraries:

- Streamlit
- scikit-learn
- PyPDF2
- spaCy
- Sentence Transformers
- Qdrant Client
- NLTK

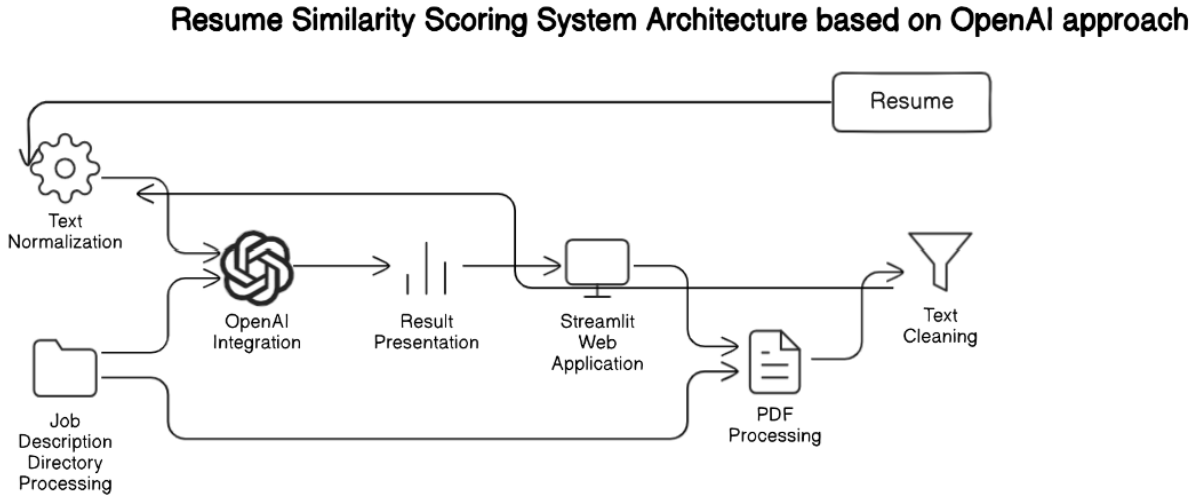
Databases:

- Qdrant

Tools/Platforms:

- GitHub
- Visual Code
- Streamlit

- **Architecture/Design:**



1. **User Interface (Streamlit App)**
    - **Functionality**: Allows users to upload a resume in PDF format.
    - **Error Handling**: Checks the file size and displays appropriate error messages.
2. **Backend Processing**
    - **PDF Processing**: Uses PyPDF2 to extract text from the uploaded resume PDF file.
    - **Text Cleaning**: Utilizes regex and string manipulation to clean the extracted text (removes punctuation, emails, URLs, non-alphabetic characters, and stopwords).
    - **Text Normalization**: Lemmatizes the cleaned text using spaCy's `en_core_web_sm` model.
3. **OpenAI Integration**
    - **API Key**: Uses the OpenAI API key for similarity scoring.
    - **Similarity Calculation**: Sends the normalized resume text and each job description text to OpenAI for similarity scoring using the GPT-3.5-turbo model.
    - **Score Retrieval**: Extracts the similarity score from the API response and filters for scores greater than or equal to 50%.
4. **Job Description Directory Processing**
    - **Directory Path**: Hardcoded path to the directory containing job description PDF files (`job_desc_dir`).
    - **File Iteration**: Iterates through each PDF file in the directory.

    - **PDF Processing**: Similar PDF processing as the resume for each job description file.
5. **Result Presentation**
    - **Top Matches**: Sorts and displays the top 5 job descriptions with the highest similarity scores.
    - **Output**: Displays the job description filenames and their respective similarity scores in a formatted manner.

•Development:

Technologies and Frameworks Used:

- **Programming Languages:** Python
- **Libraries/Frameworks:**
    - PyPDF2: Used for PDF file handling and text extraction.
    - spaCy (with `en_core_web_sm` model): Utilized for text processing, including lemmatization and tokenization.
    - nltk: Used for additional text processing tasks such as stopwords removal.
    - Streamlit: Used for developing the web application interface.
    - OpenAI API: Integrated for similarity scoring using GPT-3.5-turbo model.

Coding Standards and Best Practices:

- **Clean Code:** Followed principles of clean and readable code, adhering to Python's PEP 8 style guide.
- **Modular Design:** Encapsulated functionality into separate functions for clarity and maintainability.
- **Error Handling:** Implemented error handling mechanisms to catch and display errors to users using Streamlit's error messages.
- **Version Control:** Utilized version control (Git) for managing code changes, collaborating, and maintaining project history.**Challenges Encountered and Solutions:**

- **PDF Text Extraction:** Handling variations in PDF structure and formatting that affected text extraction accuracy. Resolved by testing with different PDF files and refining regex patterns for text cleaning.
- **API Integration:** Initial issues with API connectivity and response parsing. Addressed by debugging API request payloads and ensuring correct data formatting before sending requests.

- **Performance Optimization:** Managing large PDF files and processing times. Implemented batch processing and optimized text cleaning algorithms to improve overall performance.

Testing:

- **Unit Tests:**

  - **Text Processing:** Unit tests were conducted to validate the correctness of text extraction, cleaning, and normalization functions using sample PDF files.
  - **API Integration:** Mocked responses were used to test the integration with the OpenAI API for similarity scoring, ensuring correct data handling and response parsing.

- **Integration Tests:**

  - **End-to-End Workflow:** Integration tests verified the entire workflow from uploading PDF resumes and job descriptions to displaying similarity scores.
  - **Error Handling:** Tests were conducted to simulate error scenarios, such as invalid file uploads or API failures, to validate error messages and user feedback.

- **System Tests:**

  - **Performance Testing:** System tests focused on performance metrics, including response times for PDF processing and API calls, to ensure acceptable performance under different load conditions.
  - **Scalability:** Tested the system's ability to handle multiple concurrent users uploading and processing files simultaneously.

Deployment Process

The AI Resume Matchmaker application was deployed using a streamlined process to ensure consistency and reliability across different environments:

- **Deployment Scripts:** Utilized shell scripts for automating the deployment process. These scripts handled tasks such as environment setup, dependencies installation, and starting the Streamlit application.
- **Continuous Integration/Continuous Deployment (CI/CD):** Integrated with GitHub Actions for automated testing and deployment. This ensured that

changes pushed to the main branch were automatically deployed to production after passing all tests.
- **Environment Configuration:** Configured environment variables for sensitive information such as API keys and database credentials, ensuring security and separation of concerns.
- **Containerization:** Dockerized the application for easier deployment and portability across various cloud platforms and local environments.

Deployment Instructions

To deploy the AI Resume Matchmaker application in different environments, follow these instructions:

1. **Local Deployment:**
    - Clone the GitHub repository to your local machine.
    - Install dependencies using `pip install -r requirements.txt`.
    - Set up environment variables for API keys and other configurations.
    - Run the application using Streamlit: `streamlit run app.py`.
2. **Cloud Deployment (e.g., Heroku):**
    - Create a Heroku account and install the Heroku CLI.
    - Initialize a new Heroku app and connect it to your GitHub repository.
    - Configure environment variables in the Heroku dashboard or using the CLI.
    - Deploy the application to Heroku using Git or GitHub integration.
3. **Containerized Deployment (Docker):**
    - Build the Docker image: `docker build -t ai-resume-matchmaker .`
    - Run the Docker container: `docker run -p 8501:8501 ai-resume-matchmaker`
    - Configure environment variables using Docker environment files or `-e` flags.

User Guide Using the Application

The AI Resume Matchmaker application helps you match resumes to job descriptions efficiently. Follow these steps to use the application:

1. **Upload Resume:**
    - Click on the "Upload Resume (PDF)" button and select your resume file in PDF format.
    - Ensure the file size is within the allowed limit (200MB).

2. **View Matching Job Descriptions:**
    - Once the resume is uploaded, the application processes it to extract and normalize text.
    - Job descriptions stored in the specified directory (`sample jd`) are compared against the uploaded resume.
    - The application displays the top 5 matching job descriptions along with their similarity scores (if score >= 50%).

Troubleshooting Tips

If you encounter issues while using the AI Resume Matchmaker application, consider the following troubleshooting tips:

- **File Upload Errors:**
    - Ensure the resume file is in PDF format and within the allowed file size limit.
    - Check internet connectivity if the application fails to process uploads.
- **API Connectivity Issues:**
    - Verify that the OpenAI API key (`api_key`) is correctly configured in the environment variables.
    - Monitor API usage limits to avoid exceeding quota limits.
- **Performance Concerns:**
    - Optimize PDF processing and text normalization functions for better performance.
    - Consider upgrading system resources if processing large PDF files takes longer than expected.

Conclusion

Project Outcomes and Achievements

The AI Resume Matchmaker project successfully addressed the challenges of manual resume screening by automating the process using AI and natural language processing techniques. Key outcomes include:

- **Improved Efficiency:** Reduced time and effort required for recruiters to screen and match resumes to job descriptions.
- **Enhanced Accuracy:** Provided more accurate and objective matching of candidate skills and qualifications with job requirements.
- **User-Friendly Interface:** Streamlit-based application with intuitive user interactions and clear feedback mechanisms.**Lessons Learned and Future Improvements**

Throughout the project, several lessons were learned that can guide future enhancements:

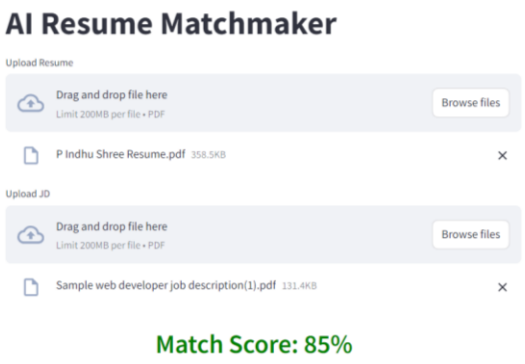
- **API Integration Challenges:** Overcame initial hurdles with API connectivity and optimized API usage for reliability and performance.
- **Scalability Considerations:** Explored containerization options for easier scaling across different environments.
- **User Feedback:** Incorporated user feedback to improve error handling and enhance user experience.

Looking forward, future iterations of the AI Resume Matchmaker could focus on:

- **Advanced AI Models:** Implementing more advanced AI models for better semantic understanding and context-aware matching.
- **Enhanced Deployment Options:** Offering deployment options via cloud services with auto-scaling capabilities.
- **Integration with ATS:** Integrating with Applicant Tracking Systems (ATS) for seamless recruitment workflows.

Appendices:

Single OpenAI Approach



Multiple JD and single Resume

Qdrant Database UI