HR ASSISTANCE SYSTEM

Problem Statement

The problem statement this project aims to solve is the time-consuming and often subjective process of manually reviewing resumes to shortlist candidates based on their skills and relevant experience. By automating this process, the HR assisting system provides a more efficient and objective approach to candidate selection.

Objective

The objective of this project is to create an HR assisting system that streamlines the hiring process by analyzing resumes/CVs and matching them with job descriptions. The system aims to identify the most suitable candidates based on their skills and qualifications.

Methodology to Solve the Problem

The proposed methodology of this project involves a multi-step process to effectively analyze resumes/CVs and match them with a job description. Here is a detailed breakdown of the methodology:

1. Resume/CV Processing: The system accepts uploaded resumes/CVs in various formats, such as PDF or Word documents. It extracts the textual content from these documents, including contact information and skills, using techniques like text extraction and parsing.

2. Job Description Input: The user provides a concise job description outlining the required skills and qualifications for the position. This information serves as the basis for matching the resumes/CVs.

3. Text Analysis: The system utilizes natural language processing (NLP) techniques to analyze both the job description and the extracted resume/CV content. The text is preprocessed to remove stop words and tokenized into individual words.

4. Skill Matching: Using cosine similarity, the system compares the job description with the skills extracted from the resumes/CVs. Cosine similarity measures the similarity between two vectors of skills. Higher similarity scores indicate a better match between the job description and a candidate's skills.

5. Candidate Selection: Based on the similarity scores, the system selects the resumes/CVs that closely match the job description. These candidates are considered to be the most suitable for the position.

6. Email Notifications: The system extracts the email addresses from the selected resumes/CVs and sends automated notifications to the candidates. The notifications inform them about their selection and provide further instructions for the HR interview process.

7. Interview Question Generation: For each selected candidate, the system generates a set of unique interview questions based on their matched skills. These questions assess the candidate's knowledge and suitability for the position.

8. Two custom modules, Sorter and ReSAn, are integrated into the system. Sorter is responsible for sorting candidate scores based on their skill matching, while ReSAn includes classes for resume analysis and interview question generation. The ResumeAnalyser class utilizes the OpenAI API to analyze resumes/CVs by matching skills with job descriptions through cosine similarity. The Interview_Chat class generates unique interview questions for each candidate based on their skills, utilizing the power of OpenAI models like "gpt-3.5-turbo-16k" and "gpt-3.5-turbo".

Scope of the Solution Proposed

The proposed solution aims to automate and streamline the candidate selection process by leveraging modern technologies and methodologies. Here are the key aspects within the scope of this solution:

1. Resume/CV Analysis: The solution includes the ability to process and extract relevant information from uploaded resumes/CVs. This includes extracting contact information, skills, and other details necessary for candidate evaluation.

2. Job Description Matching: The system compares the job description with the extracted skills from the resumes/CVs. The matching process utilizes cosine similarity to identify the best-matched candidates based on skills and qualifications.

3. Candidate Selection: The solution selects the top candidates who closely match the job description criteria. It considers the similarity scores between the job description and the candidate's skills to determine the best-fit candidates.

4. Email Notifications: The system automatically sends email notifications to the selected candidates, informing them about their selection and providing further instructions for the HR interview process.

5. Interview Question Generation: The solution generates a unique set of interview questions for each selected candidate. These questions are tailored to assess the candidate's skills and suitability for the position.

6. Interview Scheduling: The system allows HR personnel to schedule HR interviews for the selected candidates by fixing the date, time, and place. This streamlines the interview scheduling process and ensures efficient communication with the candidates.

The proposed solution focuses on automating candidate selection, streamlining the evaluation process, and optimizing HR operations. It aims to improve the efficiency and accuracy of candidate assessment and reduce the manual effort required for candidate shortlisting.

OTHER DETAILS (OPTIONAL)

The HR assisting system is built using various frameworks and libraries to streamline the candidate selection process. The user interface is developed using Streamlit, providing an interactive platform for users to upload resumes/CVs and enter job descriptions.

The system leverages the capabilities of different libraries to perform key functionalities. Base64 is used to encode and decode binary files, such as images and PDFs, allowing for display and download. The CSV library enables the reading of skills from a CSV file, providing a list of skills for skill matching with job descriptions.

Two custom modules, Sorter and ReSAn, are integrated into the system. Sorter is responsible for sorting candidate scores based on their skill matching, while ReSAn includes classes for resume analysis and interview question generation. The ResumeAnalyser class utilizes the OpenAI API to analyze resumes/CVs by matching skills with job descriptions through cosine similarity. The Interview_Chat class generates unique interview questions for each candidate based on their skills, utilizing the power of OpenAI models like "gpt-3.5-turbo-16k" and "gpt-3.5-turbo".

Data processing and manipulation tasks are handled using Pandas, allowing the system to read and extract information from the skills.csv file. NLTK and Scikit-learn are employed for text preprocessing and cosine similarity calculations. NLTK provides tokenization and stop word removal, while Scikit-learn's cosine_similarity function measures the similarity between skill vectors.

Additional libraries are utilized for various tasks. FuzzyWuzzy assists in fuzzy string matching, specifically for comparing not-met requirements from resumes/CVs with job descriptions. ReportLab facilitates the creation of PDF files, enabling the generation of interview question documents for downloading. The re library helps extract email addresses from resumes/CVs using regular expressions, and the OS library handles file system operations and path handling.

The integration of the Fitz library allows the system to extract text from uploaded PDF documents during the document_input function. Finally, the OpenAI API serves as a critical component for generating text-based responses and conversations, empowering the ResumeAnalyser and Interview_Chat classes with the capabilities of state-of-the-art language models.

Frameworks and Libraries Used in This Project

1. Streamlit: Streamlit is used as the web application framework for building the user interface of the HR assisting system. It provides an interactive and responsive interface for users to upload documents, enter job descriptions, and view the results.

2. Base64: The base64 library is utilized to encode and decode binary files, such as images and PDFs, into a format that can be embedded in HTML pages or downloaded by users.

3. CSV: The CSV library is used for reading skills from a CSV file. This file contains a list of skills for skill matching with the job description.

4. Sorter: The sorter module is a custom module that is imported and used in the main code. It includes the function "sortscores" to sort the candidates based on their skill score.

5. ReSAn: The ReSAn module is another custom module used in the main code. It includes two classes, "ResumeAnalyser" and "Interview_Chat," which perform resume analysis and interview question generation, respectively.

6. Pandas: The pandas library is used for reading and processing data from the skills.csv file. It provides data manipulation and analysis capabilities.

7. NLTK (Natural Language Toolkit): The NLTK library is used for text pre-processing tasks such as tokenization, stop word removal, and word normalization. It includes resources for tokenizers, stopwords, and other NLP functionalities.

8. Scikit-learn: Scikit-learn is a machine learning library that is used for calculating cosine similarity between skill vectors. It provides the "cosine_similarity" function for measuring the similarity between two vectors.

9. FuzzyWuzzy: The fuzzywuzzy library is used for fuzzy string matching. It includes the "fuzz.token_sort_ratio" function, which calculates the similarity between two strings based on token sorting.

10. ReportLab: ReportLab is a library used for generating PDF files. It provides functionality for creating PDF documents, adding text, and formatting the content.

11. Re (Regular Expression): The re library is used for pattern matching and extraction of email addresses from text. It includes functions like "re.findall" to find all occurrences of a pattern in a given string.

12. OS: The os library is used for file system operations and path handling. It is utilized to get the current directory path and join paths to access images and other files required in the project.

13. Fitz: Fitz is a library for interacting with PDF files. It is mainly used to read the text content from uploaded PDF documents in the document_input function.

14. OpenAI API: The OpenAI API is used for generating text-based responses and conversations. It is integrated into the ReSAn module to perform resume analysis and interview question generation tasks using OpenAI models like "gpt-3.5-turbo" and "gpt-3.5-turbo-16k".

These frameworks and libraries are used in combination to create a comprehensive HR assisting system that automates resume analysis, skill matching, interview question generation, and document processing.