AUTOMATED JOB MATCHING AND INTERVIEW ANALYSIS SYSTEM

An AI-Powered Solution for Enhancing Recruitment Efficiency through Job-Resume Matching and Candidate Interview Analysis

NOVEMBER 10, 2024

Contents

1. Problem Statement and Project Description

Objective & Goal

Objective:

The primary objective is to design an intelligent tool that enhances the recruitment process through the automation of matching resumes with job requirements and assessing interview content for detailed feedback.

Goal:

Implement a system to analyze video interviews for communication accuracy and skills, providing comprehensive feedback. This system leverages **Retrieval-Augmented Generation (RAG)** and **vector databases** to retrieve and compare interview responses with contextually relevant data for thorough and effective evaluations.

Description

The project is divided into two main modules:

- **Job Matching System**:
  - This module facilitates the seamless comparison of candidate resumes with job descriptions. It assesses how well candidates fit specific roles based on their skills, experience, education, and other relevant attributes. The system uses NLP and vector similarity to match resumes to job descriptions stored in a vector database, providing recruiters with data-driven insights.

- **Interview Analysis Module**:
  - This component automates the evaluation of video interviews to gauge candidate communication. It assesses aspects such as communication style, active listening, and engagement levels. By using AI-driven transcription and analysis tools, this module extracts key elements from interviews and generates a detailed analysis to support the recruitment decision-making process.

2. Technology and Techniques Used

Technologies Utilized:

- **Python**: Core language for developing both modules.
- **Streamlit**: Provides a user-friendly web interface.
- **Google Generative AI (gemini-1.5-flash)**: Used for content generation and interview analysis summarization.
- **Hugging Face Transformers**: Applied for ASR (automatic speech recognition) and sentiment analysis.
- **SentenceTransformers**: Employed for semantic similarity and embedding generation.
- **Pinecone**: A vector database for efficient job-resume matching.
- **Tempfile, PyMuPDF, python-docx**: Used for handling file uploads and content extraction.
- **FFmpeg**: Handles audio extraction from video files.
- **dotenv**: Loads API keys securely from environment variables.

Techniques Used:

- **Natural Language Processing (NLP)**: Extracts and processes text from resumes and job descriptions.
- **Embedding Models**: For encoding job descriptions and resumes into vector representations for similarity comparison.
- **Retrieval-Augmented Generation (RAG)**: Integrates retrieval mechanisms for justification generation.
- **Sentiment Analysis**: Analyzes the tone and sentiment of candidate responses.
- **Regex Patterns**: Extracts questions and key information from transcriptions.

3. High-Level Architecture Diagram and Workflow

Job Matching System Workflow:

- **Job Description Upload & Parsing:**
  - Recruiters upload job descriptions (TXT or JSON formats).
  - The system extracts job requirements like skills, experience, and qualifications using NLP-based regex patterns.

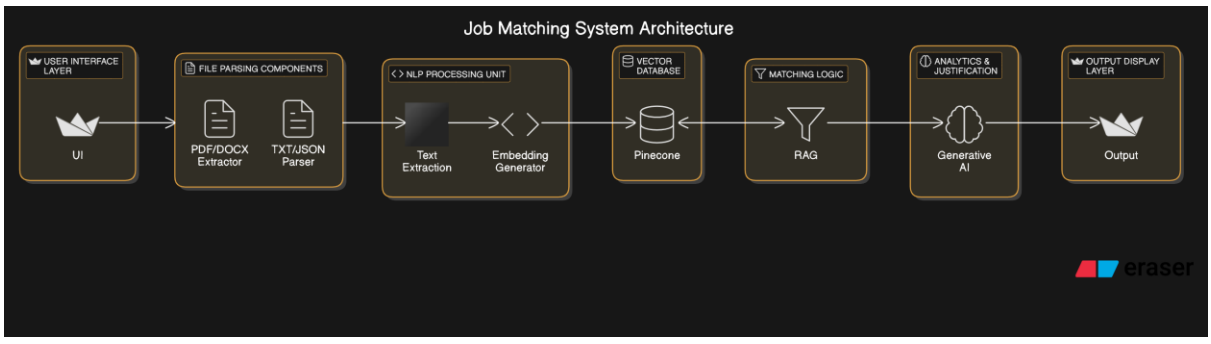
- **Resume Upload & Extraction:**
  - Supports candidate resume uploads in PDF or DOCX format.
  - Extracts information such as skills, years of experience, education, and certifications using content parsing and regex.

- **Embedding Generation:**
  - Transforms job descriptions and resumes into vector embeddings using the SentenceTransformer model for efficient similarity comparison.

- **Profile-Job Matching:**
  - Stores job description embeddings in a Pinecone vector database.
  - Matches candidates to job descriptions based on vector similarity, considering skills, experience, and qualifications.

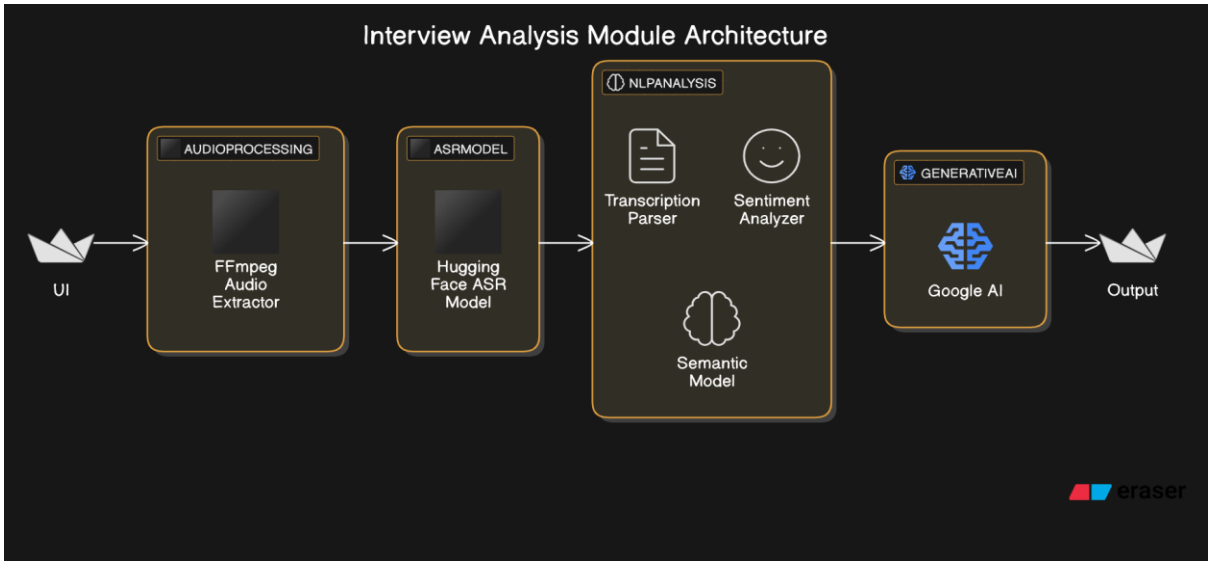
- **Match Analytics & Justification (RAG-based):**
  - Uses Retrieval-Augmented Generation (RAG) to generate justifications, drawing context from matched job descriptions and resume details.
  - Provides analytics that highlight the match between candidate and job in areas like:
    1. Skill alignment
    2. Experience match
    3. Education fit
    4. Technology compatibility

- **Generative AI for Analysis:**
  - Summarizes job descriptions and generates structured justifications using Google GenAI for clear insights on the candidate-job fit.



Interview Analysis Module Workflow:

- **Video Upload & Processing:**
    - Users upload interview videos via a Streamlit interface.
    - The module extracts audio from the video file using FFmpeg and transcribes the content using an Automatic Speech Recognition (ASR) model.
- **Interview Summarization & Analysis:**
    - Summarization:
        - A Generative AI model (Google GenAI) generates a summary, capturing key points in the interviewee's responses.
    - Communication Style Analysis:
        - Sentiment analysis evaluates the clarity, tone, and coherence of responses, providing insights into how effectively the candidate communicates.
    - Active Listening Evaluation:
        - Measures semantic similarity between questions and answers to assess the candidate's attentiveness and relevance in responses.
    - Engagement Analysis:
        - Scores engagement based on response length and conversational cues (e.g., rapport-building phrases) to assess interaction quality.



4. Process of How to Run the Code

Setup Instructions and assumption:

To effectively run the **Job Matching System** and **Interview Analysis Module**, follow these detailed setup instructions:

1. Environment Setup

- **Python Version**: Ensure Python 3.8 or higher is installed.
- **Virtual Environment (Recommended)**:

```
python -m venv env
source env/bin/activate  # On Windows: env\Scripts\activate
```

2. Install Required Dependencies

Run the following command to install all required libraries:

```
pip install streamlit ffmpeg-python python-docx PyMuPDF google-generativeai transformers sentence-transformers pinecone-client python-dotenv
```

3. Environment Variables Configuration

- Create a .env file in the root directory of your project and add the necessary API keys:

```
GOOGLE_API_KEY=<your_google_api_key>
PINECONE_API_KEY=<your_pinecone_api_key>
```

- These keys are essential for connecting to the **Google Generative AI** and **Pinecone** services.

4. Running the Streamlit Applications

Navigate to the directory containing the code modules and run the respective Streamlit applications:

- **For the Interview Analysis Module**:

```
streamlit run Interview_Analysis.py
```

- **For the Job Matching System**:

```
streamlit run Job_Matching.py
```

5. Input Format and Upload Instructions

- **Job Matching System**:
  - **Job Descriptions**: Upload files in .txt or .json formats that outline job roles and requirements.
  - **Resumes**: Upload candidate resumes in .pdf or .docx formats.
- **Interview Analysis Module**:
  - **Video Files**: Upload video files in .mp4, .avi, .mov, or .mkv formats for processing.

6. Usage Guidelines

- **Job Matching System**:
  - The system will parse and store job descriptions in a vector database.
  - Upload a resume, and the tool will analyze and display the top matching job roles based on skills, experience, and other relevant factors.
- **Interview Analysis Module**:
  - Once a video is uploaded, the system will extract audio, transcribe it, and perform an analysis.
  - The analysis includes summarizing responses and evaluating communication style, active listening, and engagement.

7. Assumptions Made

To ensure the effective operation of the **Job Matching System** and **Interview Analysis Module**, the following assumptions are made:

- **Structured Job Descriptions**: Job descriptions uploaded to the system should clearly outline key details such as required experience, essential skills, and primary responsibilities. This structured format ensures accurate extraction and matching of job requirements.

- **Formatted Resumes**: Candidate resumes must be well-organized, with distinct sections for **work experience**, **education**, **skills**, and **certifications**. Properly formatted resumes improve the accuracy of data extraction, allowing the system to effectively analyze and compare them to job requirements.

- **Interview Video Quality**: The video files uploaded for the **Interview Analysis Module** should be clear and of reasonable quality, with audible sound and minimal background noise. This ensures that the audio extraction and subsequent transcription processes are accurate, which is crucial for effective analysis of communication and engagement.

By adhering to these assumptions, the system can deliver more precise and relevant results.

5. Detailed Description of Code Workflow

Job Matching System Module:

This code provides a streamlined system for matching a candidate's resume to relevant job descriptions using embedding-based similarity searches and metadata analysis. The application is built with Streamlit for the UI, Pinecone for the vector database, Sentence Transformers for embedding generation, and Google Generative AI for content summarization and generation.

Code Workflow for Job Matching System Module

1. Imports and Environment Setup
This section imports essential packages and initializes the environment variables.

- **Libraries**: os, json, re, streamlit, numpy, dotenv, and others.
- **Environment Configuration**: The .env file is loaded using load_dotenv() to access sensitive information such as API keys.
- **Embedding Model Initialization**: The Sentence Transformer model 'all-MiniLM-L6-v2' is loaded to encode job descriptions and resumes into 384-dimensional embeddings.
- **Pinecone Initialization**: A Pinecone client is set up using the API key from environment variables.
- **Google Generative AI Initialization**: Configured with an API key to generate summaries and justifications.

2. Vector Database Structure
The Pinecone vector database is configured as follows:

- **Index Parameters**:
  - **Index Name**: job-matching
  - **Dimension**: 384 (based on the embedding model used)
  - **Metric**: Cosine similarity for measuring vector closeness.
- **Index Creation**:
  - Checks if the index exists. If not, creates it with a specified ServerlessSpec configuration for efficient querying and storage.
  - **Metadata Fields**: Each job description vector stores metadata:
    - **title**: File name of the job description.
    - **content**: Full job description text.
    - **min_experience**: Minimum required experience.
    - **skills**: Extracted relevant skills.
    - **education**: Relevant educational qualifications.
    - **certifications**: Extracted certifications related to the job.

3. Helper Functions
These functions handle file parsing, content extraction, and metadata extraction.

- **File Content Extraction**:
  - extract_resume_content(): Determines file type (PDF or DOCX) and calls the appropriate extraction function.
  - extract_text_from_pdf(): Reads PDF content using PyMuPDF.
  - extract_text_from_docx(): Extracts text from DOCX files using python-docx.
- **Job Description Parsing**:
  - extract_job_description(): Reads content and structured information from job description files.
  - **Minimum Experience Extraction**:

        - extract_minimum_experience(): Uses regex to find minimum experience in years from the job text.
- **Structured Information Extraction**:
    - extract_structured_info(): Extracts **skills**, **education**, and **certifications** from content using regex patterns for commonly sought skills, degrees, and certifications.
    - extract_resume_info() and extract_job_info(): Calls extract_structured_info() to retrieve skills, education, and certifications for resumes and job descriptions.

4. Logic Functions
Core functionality for storing job descriptions, retrieving matches, and generating justifications.

- **Store Job Descriptions**:
    - store_job_descriptions(): Embeds and stores job descriptions in the Pinecone vector database.
        - **Duplicate Check**: Checks if a job description already exists in Pinecone; if it does, deletes and replaces it.
        - **Upserting**: Adds or updates each job description as a vector in the Pinecone index.
- **Matching Job Descriptions with Resume**:
    - retrieve_matching_jobs(): Uses the candidate's resume embedding to search for top job matches in Pinecone.
    - **Experience Filtering**: Filters job descriptions based on minimum experience requirements.
- **Generating Justification**:
    - generate_justification_with_rag(): Uses a Retrieval-Augmented Generation (RAG) approach.
        - **Matching Context**: Constructs a prompt using matching skills, education, and certifications for the Generative AI model.
        - **Content Generation**: Generates a structured justification explaining the match.

5. Streamlit UI Code
The user interface where users upload job descriptions and resumes to initiate the matching and analysis process.

- **UI Components**:
    - **File Uploaders**: Allows users to upload job descriptions (TXT or JSON) and a resume (PDF or DOCX).
- **Process Flow**:

1. **Store Job Descriptions**: Calls store_job_descriptions() to store job descriptions in the Pinecone vector database.
2. **Resume Content Extraction**: Extracts resume content and calculates the candidate’s experience.
3. **Matching Jobs Retrieval**: Calls retrieve_matching_jobs() to get top matching jobs.
4. **Result Display**:
    - Displays the best-matching job with similarity scores and experience requirements.
    - Summarizes the job description using summarize_text().
    - Generates a justification using generate_justification_with_rag().
    - Displays matched skills, education, and certifications.

- **No Match Warning**: If no suitable match is found, shows a warning message.

Interview Analysis Module:

This code workflow provides a structured approach to analyzing interview transcripts by leveraging multiple AI tools and models, and finally, presents the results in an intuitive Streamlit UI. This modular setup allows easy updates to each section if additional features or analyses are needed in the future.

Code Workflow for Interview Analysis Module

1. **Imports and Environment Setup**:
    - Import necessary libraries:
        - streamlit for UI components.
        - ffmpeg for audio extraction from video.
        - tempfile and os for file handling.
        - google.generativeai and transformers pipelines for AI functionality.
        - SentenceTransformer from sentence_transformers for semantic similarity.
        - dotenv for loading API keys.
    - Load environment variables with load_dotenv() to access the Google Generative AI API key.

2. **Configuration of API and Model Instances**:
    - Configure Google Generative AI by setting up genai with the API key obtained from environment variables.
    - Initialize Hugging Face's ASR model (openai/whisper-base) for audio-to-text transcription.
    - Initialize the SentenceTransformer model (paraphrase-MiniLM-L6-v2) for semantic similarity between questions and answers.

3. **Helper Functions**:
    - **extract_audio(video_path, audio_output_path)**:
        - Uses ffmpeg to extract audio from the uploaded video file, saving it in .mp3 format.
        - Returns the path to the extracted audio file for later processing.
    - **transcribe_audio(audio_output_path)**:
        - Transcribes audio into text using Hugging Face's asr_pipeline.
        - Returns the transcription text, which will serve as the basis for analyzing the candidate's responses.
    - **extract_likely_questions(transcription)**:
        - Uses regular expressions to identify likely questions within the transcribed text.
        - Extracts and returns phrases starting with common question words (e.g., "What", "How", "Why") to filter for possible interview questions.
    - **extract_likely_answers(transcription, likely_questions)**:
        - Identifies answers by locating sentences in close proximity to questions.
        - Extracts sentences following each identified question and returns these as a dictionary of question-answer pairs.

4. **Analysis Functions**:
    - **analyze_communication_style(transcription)**:
        - Performs sentiment analysis on each answer to assess communication style.
        - Summarizes the candidate's communication by calculating an average sentiment score. An average score above 0.5 is deemed "Positive," otherwise "Negative."
        - Returns both the average sentiment score and the qualitative description ("Positive"/"Negative") of the candidate's communication style.
    - **analyze_active_listening(transcription)**:
        - Calculates the candidate's active listening by computing the semantic similarity between each question and answer.
        - Uses SentenceTransformer to embed the text and measure cosine similarity between question-answer pairs.
        - Returns an overall active listening score based on the average of these similarity measures.
    - **analyze_engagement(transcription)**:
        - Measures engagement by assessing both answer length and presence of engagement cues (e.g., "I see," "Right").
        - Counts answers with more than 10 words as indicators of engagement and adds points for each detected engagement phrase.
        - Returns an engagement score based on the sum of these counts.

5. **Generative AI Analysis**:
    - **analyze_transcript_with_scores()**:
        - Summarizes and analyzes the interview transcript using Google Generative AI, considering communication, active listening, and engagement scores as contextual inputs.
        - The function generates a prompt that instructs the AI to analyze and summarize the interview without mentioning the numeric scores.
        - The AI output includes a concise summary of the candidate's responses, evaluation of communication style, assessment of active listening, and analysis of engagement with the interviewer.
        - Returns the final text analysis produced by Google Generative AI.

6. **Streamlit UI Components**:
    - **File Upload and Processing**:
        - Displays a file uploader in the Streamlit UI, accepting video files in mp4, avi, mov, or mkv formats.
        - Upon uploading, the video file is saved temporarily and processed for audio extraction and transcription.
    - **Analysis and Display of Results**:
        - Extracts audio and transcribes it to text.
        - Calculates scores for communication, active listening, and engagement by invoking the respective analysis functions.
        - Generates a detailed interview analysis summary using Google Generative AI based on these scores.
        - Displays the transcription and AI-generated analysis output in the UI for the user.

Analyze Function Overview

This section includes functions designed to evaluate a candidate's **communication style**, **active listening**, and **engagement** based on the interview transcript. These assessments leverage general AI and NLP models to provide insight into the candidate's performance, focusing on their interaction quality.

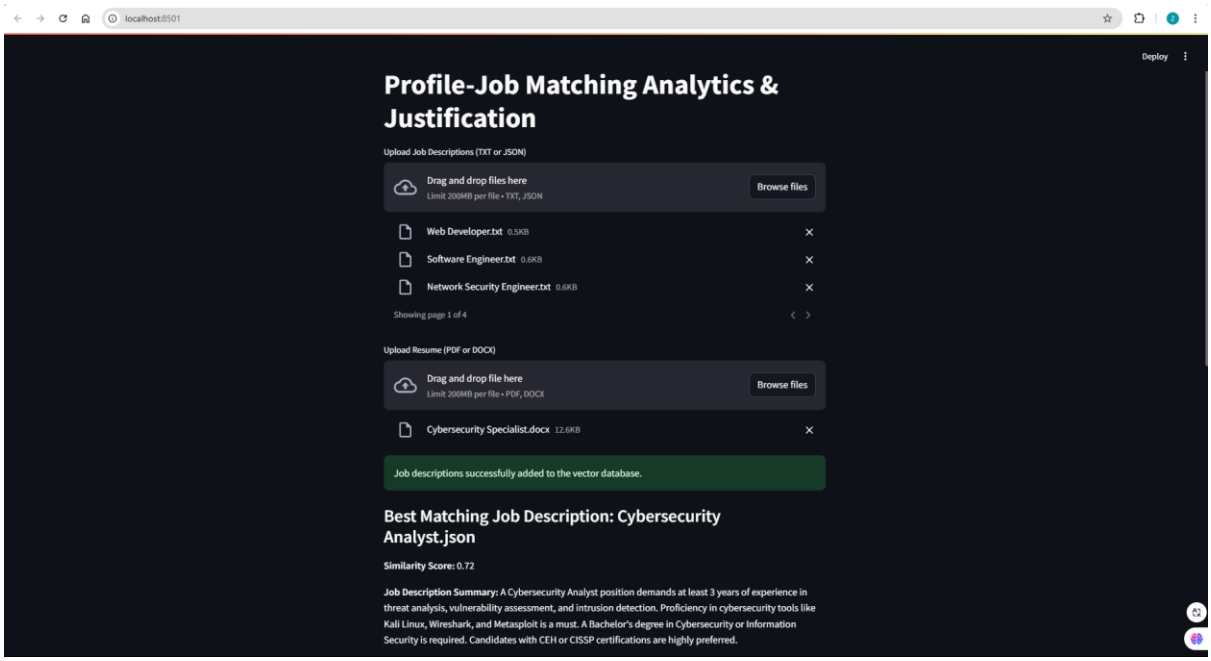
Note: The current models used (for sentiment analysis, semantic similarity, and engagement) are general-purpose and not specifically trained on interview data. As a result, the analysis provided may lack context-specific accuracy and precision for certain interview scenarios. Improved results can be obtained by fine-tuning these models with a dataset of interview transcripts, enabling them to better capture nuances in communication and engagement styles unique to interview settings.

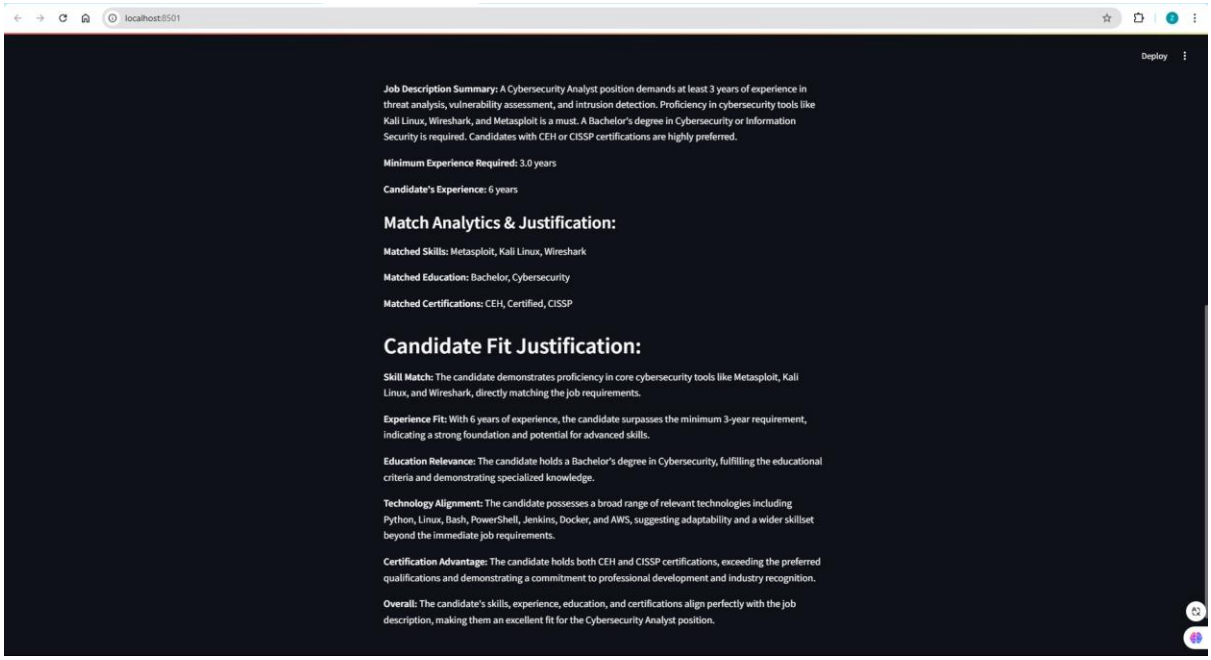
The approach aims to provide a meaningful initial analysis while guiding the generative AI prompt to address relevant aspects of communication, listening, and engagement, ultimately offering valuable insights into the candidate's performance.

6. Output Explanation

Job Matching Module Output:

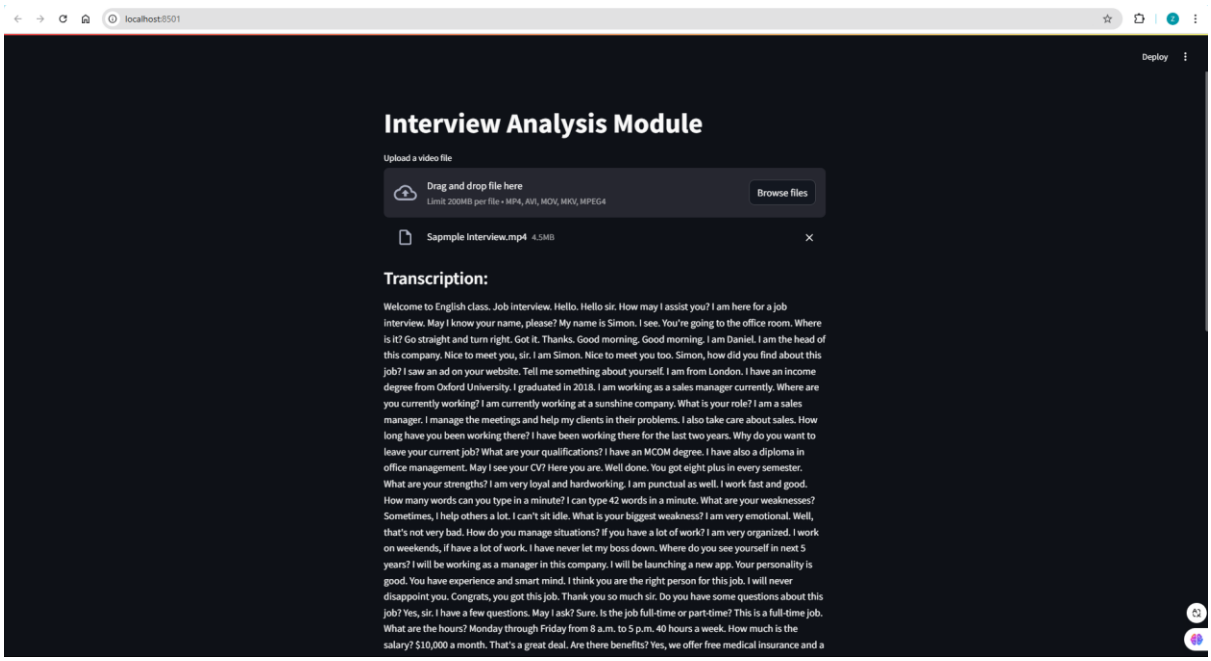
1. **Job Match List**:
   - **Top Matching Jobs**: Displays a ranked list of job positions that align well with the candidate's profile. Each job is accompanied by a similarity score, indicating the degree of match between the candidate's qualifications and job requirements.
   - **Similarity Score**: A numerical score (usually ranging from 0 to 1) representing the strength of the match. Higher scores signify a closer alignment between the candidate's skills, experience, and the job description.
2. **Justification**:
   - **Match Explanation**: Provides a detailed explanation of why each job is a good fit. It highlights specific qualifications, skills, education, and experience that align with the job's requirements.
   - **Relevant Skills and Experience**: Lists key skills and experiences from the candidate's profile that are particularly relevant to the job role, reinforcing the rationale behind each job match.

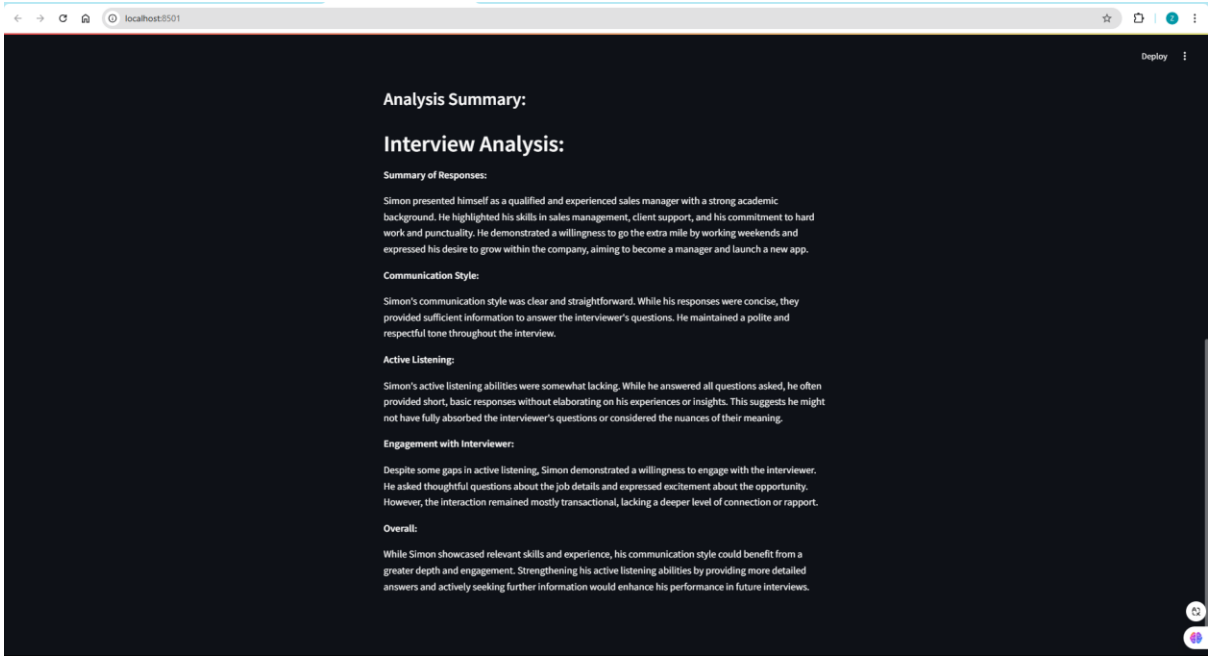




Interview Analysis Module Output:

1. **Transcription Display**:
   - Provides the full transcription of the interview in text format. This output allows the user to review the candidate's responses to each question in a readable, searchable format.
   - The transcription is generated from the audio extracted from the uploaded interview video file, ensuring an accurate representation of the candidate's spoken words.
2. **Analysis Report**:
   - **Summary**: Offers a concise summary of the candidate's answers, identifying key points and insights drawn from their responses.
   - **Communication Style**: Evaluates the tone, clarity, and overall effectiveness of the candidate's communication style. It comments on how well the candidate expresses ideas and engages with the interviewer.
   - **Active Listening**: Assesses the candidate's ability to listen attentively and respond appropriately. This section highlights the relevance and coherence of their answers, indicating whether they demonstrated effective listening.
   - **Engagement Level**: Analyzes the degree of interaction between the candidate and the interviewer, noting signs of attentiveness, rapport, and conversational cues. It measures engagement based on response length, use of conversational prompts, and feedback indicators.


localhost:8501
Deploy
Interview Analysis Module
Upload a video file
Drag and drop file here
Limit 200MB per file • MP4, AVI, MOV, MKV, MPEG4
Browse files
Sapmple Interview.mp4 4.5MB
Transcription:
Welcome to English class. Job interview. Hello. Hello sir. How may I assist you? I am here for a job interview. May I know your name, please? My name is Simon. I see. You're going to the office room. Where is it? Go straight and turn right. Got it. Thanks. Good morning. Good morning. I am Daniel. I am the head of this company. Nice to meet you, sir. I am Simon. Nice to meet you too. Simon, how did you find about this job? I saw an ad on your website. Tell me something about yourself. I am from London. I have an income degree from Oxford University. I graduated in 2018. I am working as a sales manager currently. Where are you currently working? I am currently working at a sunshine company. What is your role? I am a sales manager. I manage the meetings and help my clients in their problems. I also take care about sales. How long have you been working there? I have been working there for the last two years. Why do you want to leave your current job? What are your qualifications? I have an MCOM degree. I have also a diploma in office management. May I see your CV? Here you are. Well done. You got eight plus in every semester. What are your strengths? I am very loyal and hardworking. I am punctual as well. I work fast and good. How many words can you type in a minute? I can type 42 words in a minute. What are your weaknesses? Sometimes, I help others a lot. I can't sit idle. What is your biggest weakness? I am very emotional. Well, that's not very bad. How do you manage situations? If you have a lot of work? I am very organized. I work on weekends, if have a lot of work. I have never let my boss down. Where do you see yourself in next 5 years? I will be working as a manager in this company. I will be launching a new app. Your personality is good. You have experience and smart mind. I think you are the right person for this job. I will never disappoint you. Congrats, you got this job. Thank you so much sir. Do you have some questions about this job? Yes, sir. I have a few questions. May I ask? Sure. Is the job full-time or part-time? This is a full-time job. What are the hours? Monday through Friday from 8 a.m. to 5 p.m. 40 hours a week. How much is the salary? $10,000 a month. That's a great deal. Are there benefits? Yes, we offer free medical insurance and a


localhost:8501
Deploy
Analysis Summary:
Interview Analysis:
Summary of Responses:
Simon presented himself as a qualified and experienced sales manager with a strong academic background. He highlighted his skills in sales management, client support, and his commitment to hard work and punctuality. He demonstrated a willingness to go the extra mile by working weekends and expressed his desire to grow within the company, aiming to become a manager and launch a new app.
Communication Style:
Simon's communication style was clear and straightforward. While his responses were concise, they provided sufficient information to answer the interviewer's questions. He maintained a polite and respectful tone throughout the interview.
Active Listening:
Simon's active listening abilities were somewhat lacking. While he answered all questions asked, he often provided short, basic responses without elaborating on his experiences or insights. This suggests he might not have fully absorbed the interviewer's questions or considered the nuances of their meaning.
Engagement with Interviewer:
Despite some gaps in active listening, Simon demonstrated a willingness to engage with the interviewer. He asked thoughtful questions about the job details and expressed excitement about the opportunity. However, the interaction remained mostly transactional, lacking a deeper level of connection or rapport.
Overall:
While Simon showcased relevant skills and experience, his communication style could benefit from a greater depth and engagement. Strengthening his active listening abilities by providing more detailed answers and actively seeking further information would enhance his performance in future interviews.

7. Conclusion and Future Enhancements

Overall Project Conclusion

This project has successfully established a robust framework for automating job-candidate matching and evaluating interview performance. By integrating cutting-edge AI tools, the system enhances hiring processes, reduces human effort in initial screenings, and provides insightful analysis into candidate compatibility and communication skills.

Job Matching Module

1. **Conclusion**:

   The Job Matching Module offers a streamlined solution for matching candidate profiles with job descriptions, using advanced AI tools for data retrieval, text summarization, and justification generation. Leveraging Pinecone for efficient vector storage and retrieval, this module matches candidates based on essential criteria such as skills, experience, and education. The result is a dynamic and user-friendly system for analyzing compatibility with available job roles.

2. **Future Enhancements**:
   - **Enhanced Matching Criteria**: Add more filters, such as location, salary range, and job type, to improve result specificity.
   - **Feedback Loop**: Implement mechanisms for iterative feedback to refine and enhance job matching accuracy.
   - **Broader AI Integration**: Integrate more advanced AI models to provide comprehensive analysis and insights.
   - **Scalability**: Further optimize for high scalability and real-time matching to support large-scale applications.
   - **User Testing and Feedback**: Conduct user testing to assess system effectiveness, gathering insights for additional refinements.
   - **Expanded Use Cases**: Explore additional scenarios and applications for the profile-job matching system.

Interview Analysis Module

1. **Conclusion**:

   The Interview Analysis Module provides a foundational approach for automated interview performance assessment. By transcribing audio, analyzing communication style, assessing active listening skills, and evaluating engagement, the module offers valuable insights into a candidate's interview behavior. Google Generative AI further enhances the analysis by generating a coherent summary, making it a valuable tool for interview evaluations.

2. **Future Enhancements**:
   - **Model Training with Interview Data**: Train the AI models on specific interview data sets to improve the accuracy of insights and relevance of analyses.
   - **Sentiment Analysis and Keyword Extraction**: Add features such as sentiment analysis, keyword extraction, and summarization to deliver deeper insights into candidate responses.
   - **Platform Integration**: Integrate this module within a larger application for comprehensive automated interview analysis and feedback generation.
   - **Performance Optimization**: Enhance performance to handle high data volumes efficiently, supporting real-time analysis.
   - **Advanced Analysis Metrics**: Add metrics such as fluency, body language interpretation, and responsiveness to provide a fuller picture of interview performance.

8. References

Job Matching Module References

- Pinecone Documentation: Documentation for Pinecone, a vector database used for efficient job matching.
- Sentence Transformers Documentation: Guide for Sentence Transformers, used for encoding text for similarity analysis.
- Google Generative AI Documentation: Documentation on Google's generative AI, utilized for text summarization and analysis.
- Streamlit Documentation: Information on Streamlit, the web application framework used for the UI.
- PyMuPDF Documentation: Details for PyMuPDF, used to process PDF files in job matching.
- python-docx Documentation: Guide for the python-docx library, used to handle Word document inputs.
- dotenv Documentation: Details on the dotenv library, used to manage environment variables securely.

Interview Analysis Module References

- Google Generative AI Documentation: Reference for Google Generative AI, applied for generating analysis summaries.
- Hugging Face Transformers Documentation: Guide for Hugging Face models, used in transcription and sentiment analysis.
- Sentence Transformers Documentation: Used to measure similarity for assessing listening and engagement.
- Streamlit Documentation: Details for Streamlit, the web framework for UI in the interview analysis module.
- FFmpeg Documentation: Reference for FFmpeg, used to extract audio from video files.
- OpenAI Whisper ASR Pipeline Documentation: Details for the Whisper model, used for automatic speech recognition.
- DistilBERT Sentiment Analysis Documentation: Guide for DistilBERT, used for sentiment analysis of responses.
- Paraphrase-MiniLM-L6-v2 Documentation: Information on this model, used to assess paraphrasing and similarity in listening.
- dotenv Documentation: Used to load environment variables securely for API keys.