JobAssistAI – LLM-Powered Conversational Job Recommendation Chatbot

Abstract

JobAssistAI is an intelligent, interactive chatbot system designed to assist job seekers by recommending the most suitable job openings based on their personal and professional profiles. Built using OpenAI's GPT-3.5 Turbo language model, the system captures user input in natural language and processes it through structured prompt engineering, NLP pipelines, and contextual memory to extract essential information such as education level, work experience, technical skills, preferred job locations, and expected salary. These user preferences are matched against a curated dataset of real-world job postings to generate top 5 job recommendations in real time. The system applies LLM-based parsing, set-based skill comparisons, and layered scoring logic to ensure accurate and relevant recommendations. This report outlines the technical methodology, data preparation steps, architectural design, evaluation strategy, and key results achieved.

Introduction

In today’s dynamic job market, finding the right opportunity that matches one's qualifications and preferences can be daunting. Traditional job portals often lack personalized interaction, and users must manually sift through listings. JobAssistAI bridges this gap by utilizing the capabilities of GPT-3.5 Turbo to create a dialogue-based job assistant that not only converses naturally but also performs intelligent filtering and ranking of jobs tailored to individual profiles.

The project aims to simplify the job search process by combining conversational AI with structured data matching. Through multiple layers of understanding and validation, the chatbot ensures high-quality, user-specific recommendations, increasing the chances of candidate-job alignment.

Dataset & Preprocessing

The primary dataset used comprises 500 realistic job postings with attributes such as:

- Job Title
- Company Name
- Job Location

- Job Description
- Normalized Salary (in USD per annum)

Key Preprocessing Steps:

1. **Normalization**: Salary values were already standardized in USD per annum, avoiding the need for conversion logic.
2. **Job Feature Extraction**: A GPT-3.5 powered parser extracted structured attributes from free-text job descriptions, including:
    - Education Level (Undergraduate, Graduate, Post Graduate)
    - Required Years of Experience
    - Technical Skills

These attributes were stored as nested dictionaries in a new column called Job Features.

3. **String Cleanup**: Skills and locations were tokenized, trimmed, converted to lowercase, and stored as Python sets for efficient comparison.

System Architecture

The architecture follows a three-stage interaction framework:

Stage 1: User Intent Clarification

- **Function**: initialize_conversation()
    - Prompts the user to share their profile details interactively.
    - GPT model guides the user through questions about education, experience, skills, location, and salary.
- **Function**: get_chat_completions()
    - Responsible for interacting with OpenAI's GPT-3.5 Turbo model via the API.
    - It sends the conversation history (system prompt + user messages + assistant responses) and retrieves a fresh model-generated assistant reply
- **Function**: moderation_check()

    - Ensures safety and appropriateness of both user and assistant messages by checking them against OpenAI's content moderation system.

- **Function**: intent_confirmation_layer()
    - Validates whether all required fields are correctly provided.
- **Function**: dictionary_present()
    - Extracts the user's structured profile in dictionary format.
- **Function**: product_map_layer()
    - Extracts structured job-related features from the raw Job Description text using GPT-3.5 Turbo..

Stage 2: Job Matching Engine

- **Function**: compare_jobs_with_user()
    - Compares user profile with job dataset.
    - Scoring based on:
        - Education Match (mapped to numerical scale)
        - Experience Match
        - Skills Overlap
        - Location Intersection
        - Salary Threshold
    - Ranks jobs by score and salary in descending order.

Stage 3: Jobs Recommendation Engine

- **Function**: recommendation_validation()
    - Validates the top 5 job recommendations obtained based on the score.
- **Function**: recommendation_validation()
    - Sorts the json data obtained with score more than 1.
- Function: initialize_conv_reco()

- Generate a friendly, detailed natural-language response that presents those job recommendations to the user

Master Chatbot Controller

- **Function**: dialogue_mgmt_system()
  - Executes all the above 3 stages
  - Generates human-readable job recommendations using LLM.

Model & Prompt Engineering

The system leverages GPT-3.5 Turbo through the OpenAI API to handle both:

- **Natural Language Understanding**: Capturing unstructured user input.
- **Information Extraction & Reasoning**: Converting narrative responses into structured JSON objects.

Carefully designed prompts ensure consistency across:

- Education inference (e.g., mapping "B.Tech" to "Graduate")
- Experience quantification (e.g., "4.5 years" → 4.5)
- Skill recognition
- Location disambiguation

The chatbot uses memory context to retain previous interactions, providing a coherent multi-turn conversation.

Evaluation & Results

The chatbot was tested with multiple user profiles. In a typical case:

- Input: Graduate user with 4.5 years' experience, skills in Java and React, preferred locations in the US, expected salary $110,000.
- Output: 3–5 job recommendations matching skills, experience, and location; sorted by descending salary.

Performance observations:

- High-quality recommendations with accurate alignment to user inputs.
- Ability to handle flexible inputs (e.g., "B.Tech" → Graduate).
- In cases where not enough jobs met the criteria, fallback logic ensured partial matches were returned.

Conclusion

JobAssistAI demonstrates the powerful synergy of large language models and structured filtering logic to solve a real-world problem: personalized job search. The system successfully parses human inputs, extracts relevant attributes, compares them with a realistic dataset, and provides tailored job suggestions in natural language.

The modular architecture, reusable components, and API-driven design make the system adaptable and scalable for future extensions, such as multilingual support or dynamic job feed integration.

Future Work

- Real-time job posting integration via APIs (e.g., LinkedIn, Indeed)
- Multilingual user support for global applicability
- Voice assistant integration for accessibility
- Extended matching logic using embeddings and cosine similarity
- Feedback learning loop to personalize recommendations over time

Appendix

- Dataset Sample: 500 realistic job postings
- Model Used: GPT-3.5 Turbo
- Libraries: pandas, openai, tenacity, json, ast