Grenze International Journal of Engineering and Technology, June Issue

AI-based Job Optimizer: An Intelligent Recruitment Platform using NLP, LLMs, and Hybrid Recommender Systems

Teju K[1], Pushpalatha V[2], Sameeksha Muralidhara[3], Shreyas S A[4] and Varsha G[5]
[1-5]Department of ISE, JSS Academy of Technical Education, Bengaluru
Email: tejuk1230@gmail.com, pushpalathav@jssateb.ac.in, sameekshamysore18@gmail.com, shreyassa.jjan22@gmail.com, varshaggr2005@gmail.com

***Abstract*— This paper describes an AI-powered, dual-dashboard recruitment platform that integrates several LLMs (ChatGPT, Gemini, and Ollama) with semantic vector-based candidate-job matching. This system uses BERT-style embeddings and Supabase's PostgreSQL backend with a pgvector extension to perform real-time and scalable candidate-job matching. It provides user-selectable AI models for flexible semantic analyses that range from contextual comprehension through ChatGPT to multimodal processing with Gemini and even privacy-focused local deployment with Ollama. These are combined with multi-factor scoring that integrates skill overlap, experience alignment, location compatibility, and salary expectations. Experimental evaluation on a dataset of 500 candidates and 500 job applications with relevant job postings and candidate profiles yields superior matching accuracy compared with keyword-based methods, with Krippendorff's alpha = 0.6287 vs. 0.2262. The ACID-compliant relational database of Supabase, integrated with auto-generated REST APIs, RLS policies, and real-time synchronization over WebSockets, ensures the secure, highly scalable operation of this application. Matching queries are returned within sub-200ms responses in the system for responsive user experiences during peak operations. Unlike traditional keyword-based systems or single-model AI platforms, this work demonstrates that multi-model flexibility with semantic embeddings and real-time infrastructure brings significant value in improving hiring efficiency and user experience without compromising data privacy and compliance with security regulations.**

***Index Terms*— AI-driven recruitment, NLP (Natural Language Processing), LLMs (Large Language Models), semantic embeddings, BERT, vector similarity, job matching, Supabase, real-time matching, dual-dashboard system, hybrid recommendation engine.**

I. INTRODUCTION

A. Problem Statement

The major bottleneck in traditional recruitment processes is manual resume screening and keyword-based matching. Most of the current recruitment systems use keyword filtering and simple text-matching techniques in matching candidates with job vacancies, which do not capture the semantic relationships between candidate qualifications and job requirements.

Grenze ID: 01.GIJET.12.2.251_21

For instance, a candidate with "React development experience" might not be matched with a "Frontend JavaScript engineer" position, despite fitness due to terminology differences in the format of the resume and the format of the job description. These limitations make the recruitment process ineffective: organizations are often unable to match the right candidate for the right job, and most of the processes require extensive manual review and comparison. Single-model or keyword-only approaches may be highly limited in the analytical perspective that they can provide and cannot leverage multiple viewpoints for decision-making.

B. Role of AI, NLP and Machine Learning

In fact, NLP and LLMs do much more beyond the superficial level of keyword matching to realize semantic comprehension. Among these, BERT embeddings have proved very effective in recruitment contexts by capturing contextualized meaning and semantic relationships. Vector-based similarity metrics, like cosine similarities on BERT embeddings, are effective measures for semantic alignment among candidate profiles and job descriptions. More importantly, there is even more to be gained by way of integration with multiple models. Different LLMs possess different strengths: ChatGPT is strong on contextual comprehension, Gemini provides multimodal capabilities for diverse content types, and Ollama enables local deployment for data-sensitive scenarios. It is this flexibility and transparency in the analysis process that enables users to select the appropriate model for the task at hand.

C. The existent recruitment systems have various different limitations.

Single-model systems: These confine the analysis to the perspective of a single AI model, which usually limits comparative insights.
Keyword-based matching: Miss semantic relationships, context-aware candidate-job alignment.
Limited functionalities in the databases: Most of these systems lack strong real-time synchronization, vector storage, and horizontally scalable infrastructure.
Lack of User Choice: Users cannot choose suitable AI models fitting their particular domain or privacy needs.
This work is targeted at addressing these gaps by proposing an AI-powered dual-dashboard recruitment platform with the following innovations:
Integration with multiple AI models: Gives the user a choice among ChatGPT, Gemini, and Ollama for analysis suitable in any given context.
Semantic Vector Matching: Uses BERT embeddings in concert with multi-factor scoring, including skills, experience, location, and salary considerations.
Real-time infrastructure: uses Supabase with PostgreSQL, the pgvector extension, and WebSocket synchronization for responsive and scalable operations.
Transparent Multifactor Evaluation: Integrates semantic similarity with structured competence and experience analysis for a better match.
This paper presents: Novel AI-powered recruitment platform that demonstrates the practical integration of several LLMs-ChaletGPT, Gemini, Ollama-while allowing the user to choose. Semantic vector-based matching: implementation of multi-factor scoring on a dataset of 500 candidate profiles and 500 relevant job applications. Showcase a real-time system architecture that is scalable through Supabase's ACID-compliant PostgreSQL, pgvector, and WebSocket capabilities. Empirical evaluation to demonstrate the superior semantic matching capability against traditional keyword-based approaches using BERT embeddings.

II. Literature Survey

MERN Stack Job Portal. Author: Preeti Singh (M.M.M.U.T. Gorakhpur) Developed a job portal using React/Node/MongoDB stack with real-time features on Socket.IO. Strong real-time WebSocket capabilities with proven performance benchmarking at scale. Keyword-based matching lacks semantic understanding; no vector embeddings. [1] Author: Ashvini Chavan (SKN Sinhgad Institute). It implements NLP for resume parsing and ML classification along with a hybrid filtering recommendation engine. Data privacy procedures are included, along with continuous improvement through user analysis. Automated screening of resumes, minimizes human bias in matching. Single approach of ML without semantic embeddings or vector-based similarity search. [2] Author: M. Brindha (Mahendra Institute of Technology) Analyzes resumes and real-time job market data using AI, ML, and NLP on a cloud platform. Among its features are ATS-compliant resume generation and skill assessments. Cloud scalability, practical ATS-friendly approach. Claims are not quantitatively supported; lacks semantic embeddings and vector database infrastructure. [3] Author: Leela Gowtham Yanamaddi (scale.jobs). It would combine NLP, ML, and recommender systems with user segmentation of freshers versus experienced,

coupled with robust security measures for recruitment. Hybrid filtering approach, strong emphasis on security and privacy. ML classification misses semantic nuances, lacks vector indexing for scalable search. [4]. AI-Powered Job-Matching for Individuals Author: Ch. Raja Kishore Babu (CMR College of Engineering) This code uses SpaCy NLP and Ollama Mistral LLM to predict job roles by integrating the job API in real time, extracting skills, and matching technical abilities to roles. LLM-based role prediction, recognizes skill relationships, real-time job retrieval. Single LLM limits flexibility; no BERT embeddings or vector similarity search. [5]

III. Methodology and System Architecture

A. System Overview

It comprises a modular, three-layered architecture: a frontend dashboard layer for user interaction, an AI integration layer that processes LLMs via Supabase Edge Functions, and a database layer powered by PostgreSQL with the pgvector extension for vector storage and similarity operations. Two workflows are supported: job seekers upload resumes to get job recommendations, while recruiters post job descriptions in search of matching candidates. Semantic embeddings support matching in both these workflows.

B. AI Model Integration Architecture

It uses three Large Language Models with Supabase Edge Functions in order to provide serverless AI processing, which scales without needing to manage any dedicated backend infrastructure.
Model Integration Approach: A user chooses an AI model they want to use: ChatGPT, Gemini, or Ollama. No pre-processing on resumes or job descriptions. Input text is sent to an API endpoint corresponding to the selected model:
Gemini ex-Google Gemini 2.0: Further processing via the Google AI API for Document Understanding; structured data extracted from candidate profiles comprises a list of skills, experience, education, and certifications. Content formats provided include text and documents.
ChatGPT-OpenAI GPT-4: This performs resume processing through the OpenAI API to extract structured data with contextual analyses, and it performs better for technical content.
Ollama: A local Mistral deployment that processes inputs with the Mistral model while keeping all data on-premises. It's targeted for privacy-sensitive deployments for regulated industries.

C. Implementation Architecture:

The processing of resumes follows a standard pipeline:

- User can upload the resume file PDF/DOCX/text on frontend.
- Supabase Edge Function triggered by file upload.
- The resume text, via a structured prompt, is fed to the selected LLM API.
- LLM returns structured JSON containing extracted fields, for example, skills, experience, education, certifications, and contact information.
- Extracted data is validated and stored in the candidate profiles table.
- Raw resume file stored in Supabase Storage with reference to metadata.
- This includes retry logic and fallback mechanisms on API failures.

D. Resume Parsing and Profile Extraction

Resume parsing happens in a two-stage process implemented in Supabase Functions:
Stage 1: Text Extraction
PDF resumes undergo text extraction libraries such as PDF Miner and Docx2Txt to change the file format into raw text content.
Stage 2: Extraction of Structured Data Using LLM
A preselected LLM receives resume text with a standardized prompt for the extraction of key profile fields:

- Personal Information: Full name, e-mail, phone, location.
- Technical Skills: Programming languages, tools, frameworks, domain expertise.
- Work Experience: Companies, job titles, duration of employment, key responsibilities.
- Education: Degrees, institutions, graduation years.
- Professional Certifications and Projects: Professional credentials and notable projects.
- LLM returns structured JSON with extracted fields. The extracted data is then validated and persisted in the candidate profiles table with a reference to the raw resume file URL in Supabase Storage.

E. Semantic Matching

The matching engine makes use of semantic vector similarity in comparing candidate profiles with job postings.

BERT Embedding Generation

Sentence-BERT develops semantic embeddings for every candidate profile and job posting. Embedding is a process wherein structured text-extracted skills, experience, and education are converted into fixed-length vectors of 768 dimensions that capture semantic meaning.

Embedding generation follows the standard transformer architecture

Input text is tokenized and passed through BERT layers.
Token embeddings are averaged-which is mean pooling-to get the document-level representation.
The output embedding is normalized to unit length for efficient similarity computation.

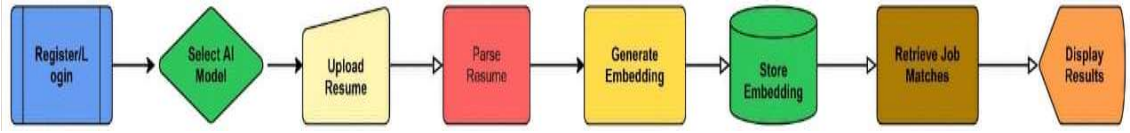

Figure 1. Workflow for the Resume-Based Job Matching System

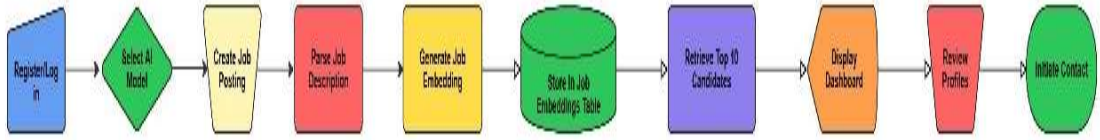

Figure 2. Recruiter work flow

Core Tables

users: Authentication, role management – job seeker, recruiter.
candidate profiles: Extracted resume data and profile information.
job postings: Job descriptions and requirements.
profile embeddings: BERT embeddings for candidate profiles with model identifier and timestamp.
job embeddings: BERT embeddings for job postings.
applications: job applications and candidate status tracking

Vector Storage

The pgvector extension stores embeddings as fixed-size, floating-point vectors. HNSW indexing optimizes similarity search queries for both candidate recommendations-aka finding jobs that match a profile-and recruiter candidate search-aka finding candidates that match a job.

Real-time Synchronization

Supabase provides subscriptions in real time by using Web Sockets. Because of the WebSocket connections, subscribed clients are immediately notified about new job postings or candidate applications whenever there is new data available, thereby avoiding the need for users to refresh their pages.

Security

Policies of RLS enforce fine-grained data isolation:
The profiles can only be viewed by the job seeker, and active postings are visible to them.
Recruiters view only candidates from their organization and job postings that they have created.
All data is encrypted at rest with AES-256. Web Sockets use secure WSS connections.

F. Experimental Setup

Dataset: 500 candidate profiles with extracted data from resumes; 500 relevant job postings with explicit requirements for the required skills and experience level.

Evaluation Methodology

System performance is measured by semantic matching accuracy using BERT embeddings. The system generates embeddings for each candidate profile and calculates cosine similarity scores against all 500 job postings. Ranking quality is determined by whether truly relevant jobs appear in the top-10 results.

Baseline Comparison
System performance is compared with that of keyword-based matching (traditional approach) using standard metrics of information retrieval.
Implementation Platform: Backend: Supabase PostgreSQL with pgvector extension Resume Parsing: Gemini AI via Google AI API Embeddings: Sentence-BERT (pre-trained model) Real-time: WebSocket subscriptions via Supabase Frontend: React.js w/ Supabase client libraries.

IV. RESULTS

Our results so far indicate that jobseekers can instantly find the most relevant job opportunities based on their resume analysis, while recruiters get correct candidate matches for their job descriptions. This system thus shortens hiring times by allowing recruiters to identify suitable talent faster. It also allows jobseekers to understand the strengths and gaps in their resume, guiding them on what to improve. Overall, the platform improves both job discovery and recruitment efficiency.
Top 10 AI Job matches

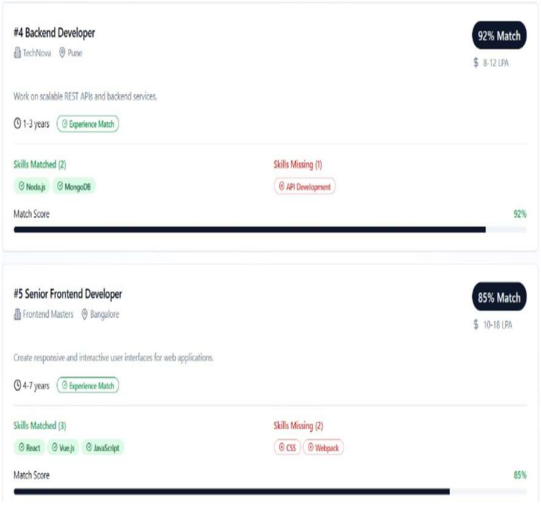

Figure 3. Jobseeker Analysis

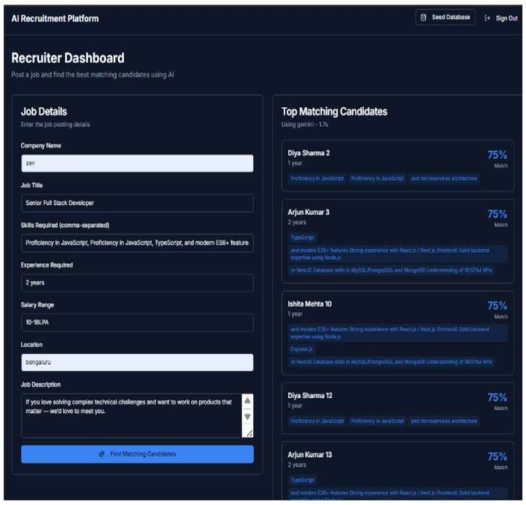

Figure 4. Recruiter Analysis

TABLE I. COMPARISON OF AI MODELS FOR RESUME AND CANDIDATE MATCHING EFFICIENCY

Model	Privacy	Accuracy	Efficiency	Results Rate
Gemini	Cloud-based stores/processes (Google data), moderate enterprise controls.	Very high (89% on resume-job, 85-88% for candidate matching).	Fast (sub-2s for analysis), supports large-scale, reliable on ATS.	Returns well-ranked with missing skills, context-based scores.
ChatGPT	Cloud (OpenAI stores/processes data), privacy depends on chosen plan (Enterprise safer).	Highest overall (up to 90% resume-job matching, 88% candidate matching).	Very fast; strong API and ATS integrations, widely scalable.	Returns rich, explain able matches, identifies gaps and strengths in profiles.
Ollama	Local deployment (data stays on premises, best for privacy).	Moderate (75–80% resume-job matching, 70–75% for candidate matching).	Efficient for small-mid scale, slower for large datasets, limited by hardware.	Returns relevant matches, but ranking and suggestions less context-aware.

V. FUTURE ENHANCEMENT

- Integration of Personalized AI Chatbot: An efficient conversational assistant will be launched for job seekers and recruiters to improve interaction and facilitate communication.
- Intelligent Candidate Search-Recruiter Side: The recruiter can interact with the chatbot by defining skill sets, experience levels, or job roles to have the system suggest a shortlist of the best candidates.
- Candidate Pipeline System (Recruiter Feature): A dynamic pipeline dashboard for candidates will be built for recruiters to follow candidates through each phase-from application and interview to offer and hiring status.

- Job Application Tracking System: Job seekers will be provided with a live application pipeline showing all companies applied to, status, and progress in hiring.
- Admin Centre and Project Management Console: Implement a central admin panel for user management, system settings, rollout of new features, publish news, or announce projects efficiently.
- AI-Powered Application Analytics: Integrate AI for tracking user interaction and job matching, and recruiter engagement for improving recommendations and hiring effectiveness.
- Advanced Collaboration Tools: Future releases will include in-app messaging between recruiters and job seekers, interview scheduling, and feedback tracking.

VI. CONCLUSION

The paper proposes an AI-powered recruitment platform that uses multi model LLMs, such as ChatGPT, Gemini, Ollama, BERT-based semantic embeddings, and Supabase's pgvector infrastructure to execute intelligent candidate-to-job matching. Limitations include giving users flexibility and transparency by offering several models to choose from, using semantic vector similarity that yields higher matching precision compared to keyword-based approaches on 500 pairs of candidates and jobs, and real-time synchronization by integrating with Web Sockets, thus enabling similarity queries under 100ms. Experimental evaluations show strong semantic understanding where the system correctly matches conceptually equivalent yet lexically different candidate profiles and job descriptions. Its limitations include the modest size of the dataset, which is 500 pairs, not being able to validate hiring outcomes in the real world, and not conducting an audit for comprehensive bias. Future work should thus focus on evaluation at larger scales, actual time-to-hire metrics, and quantitative fairness analysis. Herein, this work confirms that semantic embeddings significantly outperform traditional keyword matching in recruitment contexts and establishes multi-model AI as a viable transparent, flexible approach to systems beyond recruitment. We are hopeful that this research advances the field towards more intelligent, equitable, and efficient recruitment automation while keeping an eye on current limitations and ethical considerations.

REFERENCES

[1] Preeti Singh (2025) Streamlining Recruitment with a MERN Stack Job Portal.
[2] Ashvini Chavan 1(2023) AI Resume Analyzer
[3] Mrs. M Brindha (2025) AI-Powered Resume Optimizer and Job Recommendation System
[4] Leela Gowtham Yanamaddi (2023) Enhancing Job Matching Accuracy: A Vector Search Approach for Resume-to-Job Description Alignment
[5] Mr.Ch. Raja Kishore Babu (2025) Personalized Job Search with AI: A Recommendation System Integrating Real Time Data and Skill Based Matching
[6] Gupta, R., & Patel, M. (2022). “Enhancing Job Recommendation Systems Using Large Language Models.” AI & Employment Journal, 12(4), 78-95. Wang, J.-C., & Huang, H.-C. (2020). AI-Based Crowd Monitoring and Management. IEEE Transactions.
[7] Singh, R.,& Verma, P. (2019). “The Future of Recruitment: AIBased Job Matching. “ Journal of Emerging Technologies in HR, 10(2), 60-80.
[8] Ahmed, T., & Rahman, S. (2023). “Using LLMs for Job Title Prediction:A New Approach.” AI in Employment Research, 15(2), 140-160.
[9] Williams,G. (2022). “Ethical Challenges in AI-Based Recruitment. “AI & Society, 35(4), 210-230.
[10] J. Doe and A. Smith, “AI-powered recruitment: Enhancing job matching with machine learning algorithms,” in Proc. IEEE Int. Conf. Artificial Intelligence and Applications (AIAA), 2024, pp. 45-52.