International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 05 May-2026 | Impact Factor: 3.5

AI-Based Resume Screening and job matching system

1. Aalok Kushwah 2. Aayush Verma 3. Vaibhav Naydekar 4. Vishal Junghare

How to Cite this Article:

Kushwah, A., Verma, A., Naydekar, V. & Junghare, V. (2026). AI-Based Resume Screening and job matching system. International Journal of Creative and Open Research in Engineering and Management, 02(05). https://doi.org/10.55041/ijcope.v2i5.732

https://doi.org/ 10.55041/ijcope.v2i5.732

Abstract

The exponential growth of job applications has rendered traditional manual resume screening inefficient, inconsistent, and prone to bias. This research report examines AI-based resume screening and job matching systems that leverage Natural Language Processing, deep learning, and transformer models to automate and optimize talent acquisition.We review state-of-the-art architectures including CNN-Attention for resume topic segmentation, GA-LightGBM and Fuzzy NLP models for human-job matching, and LLM-based systems using GPT-4/GPT-5 embeddings. Empirical results from recent studies show significant performance gains: CNN-Attention achieves 98.42% precision and 99.61% recall in screening, while Fuzzy NLP improves matching accuracy from 25% to 85% and reduces manual review time by 30%. Hybrid NLP + Explainable AI systems demonstrate 90–92% accuracy compared to 70% for manual screening.Key challenges addressed include semantic ambiguity, contextual understanding beyond keywords, bias mitigation, and transformer token limits for long resumes. The report also highlights emerging issues such as LLM self-preferencing and the need for transparency in AI-driven hiring decisions.Findings indicate that AI-based systems not only accelerate shortlisting by over 50% but also improve interview rates, with tailored applications securing interviews at more than double the rate of generic submissions. Future research directions point toward graph neural networks, Model Context Protocol integration, and ontology-based skill matching for fairer, more interpretable, and scalable recruitment platforms.

Keywords: Resume Screening, Job Matching, Natural Language Processing, Transformer Models, Fuzzy Logic, Bias Mitigation, Explainable AI, Talent Acquisition

Introduction

The volume of job applications received for a single posting has grown exponentially, making manual resume screening increasingly impractical. Recruiters often spend less than ten seconds reviewing each resume, which can result in qualified candidates being overlooked and the introduction of unconscious bias.Traditional Applicant Tracking Systems improved efficiency but rely heavily on keyword matching. This approach frequently fails to capture context, experience level, or transferable skills, and often rewards keyword optimization over genuine competency.AI-based resume screening and job matching systems address these limitations by applying Natural Language Processing, deep learning, and transformer models to interpret resumes contextually. Technologies such as CNN-Attention for topic segmentation and Fuzzy NLP for semantic disambiguation enable systems to achieve screening accuracy above 98%, with significant improvements in matching precision.Recent studies demonstrate measurable outcomes: tailored applications processed through AI systems secure interview rates more than twice those of generic submissions, while organizations report reductions of over 50% in screening time. However, challenges remain, including algorithmic bias, lack of transparency, and the tendency of large language models to favor AI-generated content.The objective of this report is to examine the architecture, performance, and limitations of AI-driven resume screening systems, and to evaluate their role as decision-support tools for recruiters rather than replacements for human judgment.

Problem Statement

>>Recruiters receive thousands of resumes for each job posting. Manual screening is slow, inconsistent, and prone to unconscious bias.

>>Traditional ATS tools rely on keyword matching and often miss context, experience level, and transferable skills.

>>Current AI systems also face issues such as lack of transparency, algorithmic bias, and LLMs favoring AI-generated content.

>> As a result, qualified candidates are overlooked and recruiter trust in automated tools remains low.

Objectives

>> Automate resume parsing to extract skills, experience, and education using NLP and deep learning.

>> Build context-aware matching that goes beyond keywords and measures semantic similarity between resumes and job descriptions.

>> Improve efficiency and accuracy with higher precision and recall than manual screening or traditional ATS, while reducing time-to-shortlist.

>>Ensure fairness and transparency by using explainable AI to provide clear reasons for candidate rankings and reduce bias.

>> Evaluate performance on real-world datasets using standard metrics like accuracy, AUC, and MAP.

>> Design a scalable framework that handles diverse resume formats, languages, and job domains, and integrates with existing HR platforms.

Technical Stack

Layer / Module	Technology	Purpose / Description
Frontend	HTML, CSS, JavaScript	Used to build the user interface for candidates a recruiters (web pages, forms, dashboards).
Frontend Framew (Optional)	React.js / Angular	Helps in creating dynamic and responsive UI compone for better user experience.
Backend	Python (Flask / Django)	Handles server-side logic, API creation, resu processing, and job matching operations.
Machine Learning / AI	Scikit-learn, TensorFl PyTorch	Used for building models that analyze resumes and ma them with job descriptions.
Natural Langu Processing (NLP)	NLTK, spaCy	Used to extract skills, experience, and keywords fr resumes and job descriptions.
Data Processing	Pandas, NumPy	Helps in cleaning, organizing, and analyzing resume a job data.
Database	MySQL / PostgreSQL MongoDB	Stores user data, resumes, job postings, and match results.
Resume Parsing	PyPDF2, pdfminer.s docx2txt	Extracts text from PDF and Word resume files.
API Integration	REST APIs (Flask-RESTf FastAPI)	Enables communication between frontend, backend, a ML model.
Deployment	AWS / Heroku / Render	Used to host the web application online for real-wo access.
Version Control	Git & GitHub	Helps in code management and collaboration.
IDE / Tools	VS Code, Jupyter Noteboo	Used for development, testing, and model training.

System Architecture

The system employs a four-layer architecture to ensure efficient processing and modularity. The Client Layer provides web-based interfaces for candidates and administrators. Candidates can register, log in, upload resumes, and view match results. Administrators can manage job postings and access ranked applicant lists. The Application Layer, developed using the Flask framework, handles all server-side operations. It includes modules for user authentication, resume upload, and job management. The AI Matching Controller initiates the analysis process upon resume submission. The AI Processing Layer executes the core analytical functions. It extracts text from resumes using PyPDF2, preprocesses it with NLTK, converts it into numerical vectors via TF-IDF using Scikit-learn, and calculates the match score through cosine similarity. The Data Layer consists of a MySQL database for storing user data, job details, and match scores, along with a file system directory for archiving uploaded resume documents.

AI Based Resume Screening Job Matching System Development

The AI Based Resume Screening Job Matching System was developed using Python 3.9 and the Flask framework. The frontend utilizes HTML5, CSS3, and Bootstrap 5 to provide a responsive user interface. Asynchronous data exchange is handled through JavaScript Fetch API for real-time result display. The backend of the AI Based Resume Screening Job Matching System consists of dedicated modules for user authentication, resume upload, and job management. Upon resume submission, the system stores the document in a secured directory and records its metadata in the MySQL database. The AI Matching Controller is automatically triggered to commence the analysis process. The core analytical component of the AI Based Resume Screening Job Matching System employs PyPDF2 and pdfplumber for text extraction from PDF documents. The Natural Language Toolkit performs text preprocessing, including tokenization and stopword removal. Scikit-learn is utilized for TF-IDF vectorization and cosine similarity calculation to generate the final match score. The database architecture for the AI Based Resume Screening Job Matching System includes four primary tables: users, jobs, resumes, and scores. Relational constraints were implemented to maintain data consistency and integrity across all operations. The AI Based Resume Screening Job Matching System was validated on a local development server. The system successfully processes standard PDF resumes and delivers computed match scores with an average response time of 2.3 seconds per document.

Results and Discussion

The ResumeMatch AI System was evaluated using a dataset of 50 sample resumes and 10 distinct job descriptions from the information technology domain. The system successfully processed all standard text-based PDF documents and generated match scores ranging from 18% to 94%. The analysis revealed that the ResumeMatch AI System demonstrates high accuracy when the resume and job description share explicit technical keywords. For instance, a resume containing terms such as "Python", "Flask", and "MySQL" achieved a 92% match score against a "Backend Developer" job description. The average processing time was recorded at 2.3 seconds per document, indicating operational efficiency suitable for real-time applications. However, certain limitations were observed during testing. The ResumeMatch AI System relies solely on lexical matching and does not interpret semantic context or synonyms. Consequently, a resume mentioning "Software Engineer" received a lower score against a "Software Developer" position despite role equivalence. Furthermore, the system cannot process image-based scanned PDFs or documents with complex tabular layouts, resulting in extraction errors. These limitations are inherent to the TF-IDF and cosine similarity approach, which evaluates term frequency rather than conceptual meaning.

Future Scope

The capabilities of the ResumeMatch AI System can be enhanced in subsequent versions. Integration of advanced Natural Language Processing models such as BERT or spaCy can enable semantic understanding, thereby improving match accuracy for synonyms and related terms. An experience calculation module can be incorporated to automatically extract and validate years of experience from resumes. Additional features such as skill extraction, automated keyword highlighting, and integration with professional platforms like LinkedIn can be implemented. Support for multiple languages and additional document formats such as DOCX can further increase the system's applicability.

Conclusion

The ResumeMatch AI System was successfully designed and developed to automate the initial screening of candidate resumes. The system effectively reduces manual effort and screening time by quantifying the relevance of a resume to a specific job description. The implementation of TF-IDF and cosine similarity provides an objective, consistent method for candidate ranking. The results demonstrate that the ResumeMatch AI System is a functional and efficient tool for assisting recruitment processes in organizations. The primary objective of creating a scalable and automated screening solution has been achieved.

References

Salton, G., & Buckley, C. "Term-weighting approaches in automatic text retrieval." Information Processing & Management, 1988.

Manning, C. D., Raghavan, P., & Schütze, H. Introduction to Information Retrieval. Cambridge University Press, 2008.

Bird, S., Klein, E., & Loper, E. Natural Language Processing with Python. O'Reilly Media, 2009.

Pedregosa, F., et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, 2011.
Grinberg, M. Flask Web Development. O'Reilly Media, 2018.