AI-Driven Resume Analysis And Job-Specific Resume Optimization System

Dr. R. Aarthy[1], Dineshwar C[2], Kanvar G[3], Kishore N[4], Mohamed Yunus A[5]
[1]Assistant Professor, Dept of Artificial Intelligence and Data Science
[2, 3, 4, 5] Dept of Artificial Intelligence and Data Science
[1, 2, 3, 4, 5] Dhanalakshmi Srinivasan Engineering College (Autonomous), Perambalur – 621 212, India

Abstract- *This paper presents an AI-driven system for analyzing resumes and optimizing them to match specific job descriptions. Modern recruitment relies heavily on Applicant Tracking Systems (ATS), which filter candidates based on keyword relevance and semantic alignment. Many qualified candidates are inadvertently eliminated due to poor resume formatting or misaligned content. The proposed system employs Natural Language Processing (NLP) and Machine Learning (ML) techniques, including Named Entity Recognition (NER) for structured information extraction and Sentence-BERT (SBERT) embeddings with cosine similarity for semantic matching between resumes and job descriptions. The system identifies skill gaps, computes a quantitative match score, and provides personalized optimization recommendations including keyword suggestions, content rephrasing, and ATS formatting guidance. A Flask-based REST API enables real-time interaction. Experimental evaluation demonstrated an 88.40% matching accuracy, with 85% of users reporting improved resume quality. The system offers an accessible, intelligent, and explainable solution to bridge the gap between job seekers and automated recruitment processes.*

Keywords: Resume Analysis, Applicant Tracking System (ATS), Natural Language Processing (NLP), Sentence-BERT, Cosine Similarity, Skill Gap Analysis, Named Entity Recognition (NER), Resume Optimization.

I. INTRODUCTION

The digital transformation of recruitment has fundamentally altered how job applications are processed. Companies increasingly deploy Applicant Tracking Systems (ATS) to manage large volumes of applications, automatically parsing and ranking resumes before human review. Industry surveys indicate that over 75% of large organizations and nearly 98% of Fortune 500 companies employ such systems. These platforms reduce time-to-hire by up to 60% and cost-per-hire by 40%, yet they introduce a significant barrier: qualified candidates are routinely screened out because their resumes fail to align with ATS parsing logic, not because they lack the required competencies.

Traditional resume-writing advice emphasizes visual creativity and unique formatting, yet these elements often confuse automated parsers. Furthermore, candidates frequently submit identical resumes across diverse job postings, reducing semantic alignment with each specific role. Existing commercial solutions such asJobScan and ResumeWorded address some of these challenges through keyword density analysis, but they rely on surface-level matching, are cost-prohibitive for many users, and offer limited explainability.

This paper presents an AI-driven resume analysis and job-specific optimization system designed to overcome these limitations. Leveraging transformer-based language models and NER-driven information extraction, the system computes semantic similarity between a candidate's resume and a target job description, quantifies skill gaps, and delivers prioritized, actionable optimization recommendations. A Flask REST API provides real-time deployment, making the tool accessible through a lightweight web interface.

The key contributions of this work are: (i) a multi-format resume parser combining rule-based and NLP-based section segmentation; (ii) a semantic similarity engine using Sentence-BERT embeddings and weighted cosine similarity; (iii) a skill gap analyzer grounded in a curated skill ontology; (iv) a recommendation engine generating contextual improvement suggestions; and (v) empirical evaluation demonstrating 88.40% matching accuracy and high user satisfaction.

II. RELATED WORK

Research on automated resume processing and candidate matching has evolved through three broad phases: rule-based parsing, classical machine learning, and deep learning approaches. Each phase introduced new capabilities while exposing new limitations that motivated subsequent advances.

A. Rule-Based and Classical Approaches

Early ATS platforms relied on exact keyword matching and template-based parsing. Johnson and Williams [2] reviewed ATS evolution from the 1990s to the present, documenting false negative rates of 30–45% in keyword-based systems, meaning nearly half of qualified candidates are incorrectly filtered. These systems cannot distinguish between "software engineer" and "software developer," leading to systematic exclusion of otherwise qualified applicants.

Patel et al. [3] applied Named Entity Recognition and text classification using spaCy and NLTK to extract structured resume information, achieving reasonable accuracy on well-formatted documents but struggling with non-standard layouts. Thompson and Garcia [7] studied formatting factors affecting ATS parsing, finding that tables, graphics, and multi-column layouts cause section misidentification in over 40% of cases.

B. NLP and Embedding-Based Matching

The introduction of word embeddings marked a turning point. Mikolov et al. [14] demonstrated that Word2Vec representations capture semantic relationships, enabling similarity computation beyond exact token overlap. Chen et al. [4] proposed semantic matching using sentence embeddings and cosine similarity, demonstrating that contextual representations substantially outperform TF-IDF keyword matching for resume–job description alignment, with a 14% improvement in F1 score on their benchmark.

Devlin et al. [11] introduced BERT, whose bidirectional transformer architecture captures deep contextual relationships. Fine-tuned BERT models outperformed all prior text-only methods. However, full BERT models carry 110 million parameters, making real-time inference computationally expensive. Reimers and Gurevych[12] addressed this with Sentence-BERT (SBERT), which produces semantically meaningful fixed-length embeddings using siamese BERT networks, enabling efficient cosine similarity computation at scale. Das and Verma [20] applied transformer-based models to resume–JD matching and reported significant improvements over BERT fine-tuning baselines.

C. Skill Gap and Recommendation Systems

Rodriguez and Lee [5] explored skill gap analysis using knowledge graphs and ontologies, enabling reasoning over related skills rather than requiring exact matches. Their system identified latent skill relationships, such as inferring that proficiency in TensorFlow implies familiarity with Python, reducing false-gap reports by 22%. Miller and Singh [10] studied explainable AI approaches to recruitment, emphasizing that actionable, interpretable feedback significantly increases candidate trust and tool adoption rates.

Brown et al. [6] demonstrated that generative AI models could produce coherent resume content improvements when given structured gap analysis as input. Kim et al. [9] proposed real-time job matching using collaborative filtering, achieving sub-second recommendation latency at scale. Gupta and Sharma [18] evaluated ML-based resume screening algorithms, finding that ensemble methods combining NER with TF-IDF outperformed single-method baselines by up to 12%. Despite these advances, no freely available system integrates robust multi-format parsing, semantic matching, skill gap detection, and actionable recommendations within a unified real-time pipeline. The present work addresses this gap.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system follows a five-stage pipeline: resume parsing, job description analysis, semantic similarity computation, skill gap identification, and recommendation generation. A Flask-based REST API serves as the deployment interface, enabling real-time browser access without heavyweight infrastructure.

A. Resume Parsing Module

The Resume Parsing Module accepts files in PDF, DOCX, and plain-text formats. For PDF files, pdfplumber is applied to preserve layout and table structure; python-docx handles DOCX files. Raw text undergoes a preprocessing pipeline comprising whitespace normalization, special-character removal, and section segmentation. Section boundaries are detected using a combination of rule-based pattern matching on common headers (e.g., "WORK EXPERIENCE", "EDUCATION", "SKILLS") and a trained text classifier for non-standard layouts.

Named Entity Recognition is applied using spaCy's pre-trained model, augmented with a custom skill ontology of over 500 technical and soft skills. The module extracts contact information, professional summary, work experience entries, educational qualifications, technical and soft skills, and certifications. Output is a structured JSON object serving as input to all downstream modules.

B. Job Description Analyzer Module

The Job Description Analyzer processes raw job posting text to extract required skills, preferred qualifications, years of experience, and role responsibilities. Requirement classification distinguishes mandatory from preferred criteria

using keyword cues such as "required", "must have", and "preferred". TF-IDF scoring assigns importance weights to extracted terms, with positional weighting applied such that skills mentioned in earlier sections receive higher weight. The output is a structured representation of job requirements used in all subsequent matching and gap analysis steps.

C. Semantic Similarity Matching Engine

The core matching engine employs Sentence-BERT (all-MiniLM-L6-v2) to generate fixed-length semantic embeddings for resume sections and job description content. Cosine similarity is computed between corresponding section pairs. The final match score aggregates section-level similarities using empirically determined weights: skills (35%), work experience (35%), professional summary (15%), and education (15%).

Unlike keyword matching, semantic embeddings capture synonymy and conceptual relatedness. A resume describing "led cross-functional teams" is correctly matched to a job requirement for "team leadership experience"; similarly, "machine learning" is semantically associated with "artificial intelligence" without requiring identical terminology. This eliminates the primary failure mode of keyword-only systems. [4, 11]

D. Skill Gap Analyzer Module

The Skill Gap Analyzer performs set-theoretic comparison between the skill entities extracted from the resume and those required in the job description. Missing skills are ranked by their TF-IDF importance weight in the job description, ensuring high-priority gaps are surfaced first. Under-represented skills—present in the resume but with insufficient prominence—are detected by comparing embedding similarity of skill-section excerpts against job description skill emphasis. The CNN-style processing architecture consists of:

- Input Layer: Accepts the parsed resume JSON and job description JSON.
- Processing Layers: Computes semantic similarity scores per section.
- Aggregation Layer: Applies weighted formula to derive overall match %.
- Output Layer: Returns match score, gap list, and ranked recommendations.

E. Recommendation Engine and Deployment

The Recommendation Engine generates prioritized, personalized suggestions across four categories: (i) missing skill keywords to incorporate with suggested placement sections; (ii) content rephrasing recommendations to improve semantic alignment with job description language; (iii) action-verb substitutions to strengthen impact language (e.g., replacing "worked on" with "architected"); and (iv) ATS formatting guidelines addressing non-standard section headers, embedded tables, and graphics. Recommendations are ranked by estimated impact score. The system is deployed as a Flask REST API exposing /upload and /analyze endpoints, enabling real-time browser-based interaction through a lightweight React.js front-end interface.

IV. MODULES

Module 1 **Resume Parsing**	**Module 2** **JD Analyzer**	**Module 3** **Similarity Engine**
Extracts text from PDF, DOCX, TXT. NER identifies skills, experience, and education sections.	Processes job posting text. Classifies required vs. preferred qualifications; applies TF-IDF scoring.	SBERT embeddings + weighted cosine similarity. Section scores aggregated into overall match %.
Module 4 **Skill Gap Analyzer**	**Module 5** **Recommendation Engine**	**Module 6** **Flask REST API**
Compares resume skills vs. JD requirements. Ranks missing skills by JD importance weight.	Generates keyword suggestions, rephrasing tips, action-verb upgrades, and ATS formatting fixes.	/upload and /analyze endpoints return match score, gaps, and recommendations in real time.

V. RESULTS AND DISCUSSION

The system was evaluated using a dataset of 200 resume–job description pairs spanning five domains: software engineering, data science, marketing, finance, and healthcare IT. Ground-truth match labels were established through expert annotation by two senior HR professionals with inter-annotator agreement (Cohen's $\kappa = 0.84$). All experiments were conducted on Python 3.10 with Hugging Face Transformers and sentence-transformers libraries on an Intel Core i7 system with 16 GB RAM.

TABLE I. PERFORMANCE COMPARISON

Method	Acc.	Prec.	User Sat.
Keyword ATS	61.2%	58.4%	52%
JobScan	71.5%	69.8%	68%
TF-IDF + Cosine	74.3%	72.1%	63%
BERT (Text-only)	80.6%	79.5%	74%
Proposed (SBERT+NER)	**88.4%**	**87.2%**	**85%**

As shown in Table I, the proposed system achieves 88.40% matching accuracy, outperforming the keyword-based ATS baseline by 27.2 percentage points and the BERT text-only approach by 7.8 points. The Sentence-BERT semantic matching engine is the primary driver of this improvement, enabling correct association of semantically related but lexically distinct skill expressions. Precision of 87.20% confirms that the system's positive match predictions are highly reliable, minimizing false-positive recommendations.

Section-level analysis revealed that skills matching (35% weight) contributed most to overall accuracy, followed by experience matching. The education section showed the smallest variance across approaches, as educational credentials use standardized terminology. Summary section matching demonstrated the strongest correlation with expert annotation, suggesting professional summary alignment is a reliable proxy for overall candidate fit.

User satisfaction was assessed through a structured usability study with 40 participants. 85% rated the recommendations as beneficial, compared to 68% for the commercial JobScan baseline. Participants valued the semantic match score breakdown and the prioritized recommendation list, which reduced decision ambiguity. Qualitative feedback identified two improvement areas: parsing accuracy for multi-column PDFs and recommendation specificity for senior-level roles.

VI. BENEFITS

- Semantic Understanding: Matches synonyms and related concepts unlike keyword-only ATS, reducing false negatives for qualified candidates. [1, 2]
- Free and Accessible: Democratizes resume optimization for all job seekers including students and career changers. [1, 9]
- Personalized Recommendations: Suggestions tailored to the specific resume and job description, not generic advice.
- Multi-Format Support: Handles PDF, DOCX, and plain-text resumes with robust section boundary detection.
- Explainable Results: Match scores accompanied by section-level breakdowns, building candidate trust. [3, 9, 10]
- Cost-Effective: Eliminates the need for expensive commercial resume optimization subscriptions.

VII. CHALLENGES AND FUTURE WORK

Although promising, the proposed system faces several challenges that guide future research directions.

- Dataset Scale: A larger, more diverse annotated dataset spanning additional industries and regions is required to validate accuracy more broadly. [3, 8]
- Multi-Column PDF Parsing: Complex graphical resume layouts with side panels and floating skill bars occasionally cause section boundary errors.
- Senior-Role Specificity: Generic action-verb suggestions may be less valuable for experienced professionals who need domain-specific guidance.
- Privacy Considerations: Processing sensitive personal resume data requires strict data governance and GDPR-compliant storage policies.

Future research will focus on: (i) LLM-based auto-generation of optimized bullet points using GPT-class models; (ii) dynamic skill ontology updates via live job-portal APIs to remain current with emerging demands; (iii) cross-lingual support using multilingual BERT for non-English markets; (iv) computer vision-based layout analysis for multi-column PDF parsing; and (v) direct job-portal integration for a unified apply-and-optimize workflow. [6, 8]

VIII. CONCLUSION

This paper presented an AI-driven resume analysis and job-specific optimization system that addresses the critical gap between qualified candidates and automated recruitment screening. By combining multi-format resume parsing, Sentence-BERT semantic matching, NER-driven skill gap analysis, and a prioritized recommendation engine deployed via Flask REST API, the system provides an accessible, explainable, and effective alternative to both generic keyword matchers and expensive commercial tools. Experimental evaluation on 200 annotated resume–JD pairs demonstrated 88.40% matching accuracy and 85% user satisfaction, with consistent performance advantages across five professional domains. Future work will extend the system with LLM-based content generation, improved multi-column PDF parsing, and live skill ontology updates, further strengthening practical utility in the continuously evolving landscape of AI-driven recruitment.

REFERENCES

[1] Bhatlawande, S., Borse, R., Solanke, A., &Shilaskar, S. (2024). "**A Smart Kit Approach for Augmenting Mobility of Visually Impaired People**". IEEE Access, 12, 24659–24671.

[2] Johnson, M., & Williams, R. (2023)."**Resume Screening Systems: A Review of Automated Candidate Filtering Techniques"**. Journal of Human Resource Management, 45(3), 234–251.

[3] Patel, S., Sharma, A., & Kumar, V. (2022). "**NLP-based Resume Analysis Using Named Entity Recognition and Text Classification**". Proceedings of ICNLP, 112–125.

[4] Chen, L., Wang, Y., & Zhang, H. (2023). "**Semantic Matching of Resumes and Job Descriptions Using Sentence Embeddings**". IEEE Trans. Knowledge and Data Engineering, 35(8), 7892–7905.

[5] Rodriguez, M., & Lee, J. (2021). "**Skill Gap Analysis Using Knowledge Graphs and Ontologies**". Expert Systems with Applications, 185, 115–129.
Brown, T., Anderson, K., & Martinez, C. (2024). "**Generative AI for Resume Content Enhancement**". ACM Trans. Intelligent Systems and Technology, 15(2), 1–22.

[6] Thompson, E., & Garcia, R. (2022). ATS Compatibility: "**Resume Formatting Factors Affecting Parsing Accuracy**". Int'l Journal of Selection and Assessment, 30(4), 512–528.

[7] Nobel, S., Chen, T., & Williams, K. (2023). "**Bias Detection in AI-based Resume Screening Systems**". AI and Society, 38(3), 891–907.

[8] Kim, S., Park, J., & Lee, H. (2022). "**Real-time Job Matching Using Collaborative Filtering**". Information Processing & Management, 59(4), 102–118.

[9] Miller, T., & Singh, P. (2023). "**Explainable AI for Recruitment**". Proceedings of CHI Conference on Human Factors in Computing Systems, 1–14.

[10] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). "**BERT: Pre-training of Deep Bidirectional Transformers**". Proceedings of NAACL-HLT, 4171–4186.

[11] Reimers, N., &Gurevych, I. (2019). "**Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks**". Proceedings of EMNLP-IJCNLP, 3982–3992.