AI-Resume Coach: Instantly Match Your Resume To Jobs And Free Courses For Missing Skills

Chiluka Mounika[1],Dr.V.Kamakshi Prasad[2]
[1]Post Graduate Student, [2]Senoir Professor of CSE & Director,
M.Tech (DS),Department of Computer Science and Engineering,
Jawaharlal Nehru Technological University , Hyderabad , India
[1]Email: mounikanetha577@gmail.com
[2]Email: kamakshiprasad@jntuh.ac.in

Abstract— Before Applicant Tracking Systems (ATS) were introduced into today's recruitment process, it was necessary to understand the impact this technology has had on how resumes are assessed by humans rather than machines before getting in front of the interviewer. Comprehending how these automated filters work is a challenge for many qualified professionals, as well as fresh graduates who are unaware of how these filters react to their resume content, thereby screening out even some strong candidates because of misaligned content. The paper introduces an intelligent web-based resume evaluation platform, which connects the dots between job seekers and modern recruiting technologies, named AI-ATS. The system uses spaCy and Sentence-BERT (Sentence Transformers) as the NLP models to extract competency entities from resume documents and competency entities from target documents in the ATS, respectively calculating ATS-like cosine similarity between competency sentences. Unlike traditional "keyword" matching, AI-ATS is able to detect the contextual relationships between the skill terminologies, finding, for example, the equivalences between two different skill sets, even though they may be both semantically identical but written using different terms. The proposed architecture is a client-server design with a React frontend for providing an interactive analysis dashboard and a Python Flask backend to handle PDF text extraction with pdfplumber, NLP inferences, and a recommendation engine to map detected skill gaps to curated free learning resources. The experimental results, based on a data set of 150 resume–job-description pairs, show an average ATS score prediction alignment of 91.3% against the recruiter assessment of the CV and an F1-score of 93.7% for skill extraction. The system speeds up the resumes' self-evaluation process by lowering the average times from several hours to fewer than three minutes, and provides actionable information that is designed to promote current professional development in addition to resume screening optimization.

Keywords—Applicant Tracking System, Resume Analysis, Natural Language Processing, Sentence-BERT, Semantic Similarity, Skill Extraction, spaCy, React.js, Flask, Course Recommendation, Career Guidance.

I. INTRODUCTION

The digital transformation of recruitment has produced a landscape in which the initial gateway to employment is increasingly governed by software rather than human judgment. organizations will entrust the task of screening the documents in the initial stages to the Applicant Tracking Systems (ATS) software applications that will analyze the documents, get the structure of the documents, and provide the applicant with a list of applicants that are relevant based on a calculated relevance score. Recruitment analytics firms have long found that more than three-quarters of candidates' apps go unread before a recruiter looks at a single line of text — and that was before the social media to-do list got so overwhelming. It's not surprising to see that more than three-quarters of all submitted applications don't reach the attention of a recruiter before they're discarded, based on surveys from recruitment analytics firms, in an era when the social media to-do list has become overwhelming. This is a fundamental barrier that can hinder candidates despite a combination of technical skills and years of investment, especially given the lack of understanding about how ATS systems work.

A lack of some highly sought-after terms and phrases, a failure to use the specific terms an exact-match system recognizes, or possibly the failure to include the specific tools listed and only using umbrella terms for the tools in the advertisement, may result in a mismatch between the candidate qualifications and the ATS-derived score. A software engineer who is knowledgeable in "machine learning" may get a zero rating, as opposed to a less experienced engineer who has written each sub-framework on its own line. This gap needs a system that comprehends the context of the words that it comes into contact with, instead of counting the tokens.

This need has already been partly met by some commercial systems like Jobscan, Resume Worded, and Rezi, which make use of the keyword frequency analysis most of the time; these systems require paid levels for some of their advanced features, and they offer limited information on the score computation [2], [3]. None of the widely available platforms

has a readily accessible interface that integrates skill matching, score explanations, and learning recommendations. However, none of the widely available platforms have skill matching, score explanations, and an integrated learning recommendation combined in a readily accessible interface.

This paper introduces AI-ATS, a full-stack intelligent resume evaluation system that unifies semantic NLP-based analysis with an actionable career guidance engine. The proposed system makes the following contributions:

- A semantic skill extraction pipeline combining spaCy PhraseMatcher with Sentence-BERT embeddings that achieves 93.7% F1-score on competency identification tasks.
- An ATS-style scoring algorithm integrating lexical and semantic similarity signals to produce granular compatibility scores aligned with recruiter assessments.
- An automated recommendation engine that maps identified skill gaps to curated free online learning resources, transforming the platform from a diagnostic tool into a professional development companion.
- A decoupled React.js / Python Flask architecture enabling real-time interactive analysis with downloadable PDF reporting.

The remainder of this paper is organized as follows. Section II surveys related work in automated resume screening and NLP-based career systems. Section III describes the system architecture and key components. Section IV details the methodology for skill extraction, semantic similarity computation, and score generation. Section V presents the dataset, experimental configuration, and evaluation results. Section VI discusses implications and limitations, and Section VII concludes with directions for future research.

II. Related Work

A. ATS and Resume Optimization Platforms

The rest of this paper is organized as follows. The related work on Auto Resume Screening and NLP-based Career Systems is discussed in Section II. Here are a few of the key system components and architecture described in Section III. The methodology of skill extraction, computation of semantic similarity, and generation of the score is presented in Section IV. The data, experimental setup, and assessment results of the experiments are provided in Section V. Implications, and limitations are explored in Section VI, and Section VII outlines directions for future research.

B. Natural Language Processing for Information Extraction

What is Natural Language Processing (NLP)?B1 Introducing NLP: The Natural Language Processing (NLP) domain.

Structured information is the initial and fundamental NLP task to extract information from unstructured text documents. Honnibal et al. created a production-ready industrial-strength NLP Library called spaCy, which has a PhraseMatcher component for entity recognition in large Phrase vocabularies that are well-suited for domain-specific skill ontologies [7]. Fine-tuned NER models on resumes using resume corpora have achieved higher accuracy than 88% for technical skill extraction tasks [8]. The problem of aligning the semantics of two documents that describe different aspects of the same skill, i.e., whether two respective mentions refer to the same skill, has been solved by using dense vector representations.

Devlin et al. [9] presented the bidirectional transformer pre-training model BERT, which presents a set of word embeddings that improves the performance of the model on a variety of tasks in the area of language understanding. Reimers and Gurevych [10] proposed Sentence-BERT (SBERT), which is based on the BERT model and its modifications with siamese and triplet tasks to create embeddings for sentences that are semantically meaningful and can be used for cosine similarity comparison. Since then, SBERT embeddings have become the default benchmark in a variety of document similarity-related tasks, so that downstream document similarity applications can then calculate the semantic distance without requiring full transformer network retraining during inference.

Assistive Technology is currently considered as a career guidance and recommendation system

The marriage of NLP and recommender systems has resulted in several career-oriented platforms. Karthikeyan et al. [11] showed that the skill extraction from transformer models combined with CF systems leads to an 14% accuracy gain in job–candidate matching over a baseline method solely using keywords. In Patel et al. [12], they discussed the prediction of ATS using machine learning, finding the mean absolute error of 8.3 percentage points with recruiters' assigned scores. Neither of the studies, however, was embedded with a forward-looking component in the form of a recommendation for a skill, directing candidates to a learning path towards acquiring the skill. The AI-ATS system can overcome this by pairing the retroactive evaluation with prospective learning routes, thereby creating an all-in-one diagnostic and developmental system.

C. Career Guidance and Recommendation Systems

The intersection of NLP and recommender systems has produced several career-oriented platforms. Karthikeyan et al. [11] demonstrated that combining transformer-based skill extraction with collaborative filtering could improve job–candidate matching accuracy by 14% over keyword baselines. Patel et al. [12] explored ML-based ATS score prediction, reporting mean absolute error of 8.3 percentage points against recruiter-assigned scores. However, neither study integrated a forward-looking recommendation component that guides candidates toward skill acquisition. The AI-ATS system addresses this gap by coupling retroactive evaluation with prospective learning pathways, producing an integrated diagnostic and developmental platform.

III. System Architecture

A. High-Level Design

AI-ATS is a decoupling architecture system of client–server that separates the presentation layer from the business logic and data concerns. The React.js front end provides a single-page, interactive application, allowing users to upload their PDF resumes, add a job description, request analysis, and peruse the results on an interactive dashboard. Communication takes place through RESTful HTTP endpoints that control the backend, as operated through the Python Flask server, and supported by the Flask-CORS extension, which also permits CORS. The whole NLP computation process, including PDF parsing, tokenization, Skill extraction, generation of embeddings, similarity scoring, and retrieval of recommendations, is only done in the backend; the performance shouldn't be affected by the client's hardware capabilities.

These functional modules can be physically separated on the back end, where the processing pipeline is executed on a series of distinct functional modules that each take the input data and derive a transform, each function having a well-defined boundary. Such a modular design allows for individual units to be tested independently and eases future changes to any single unit (i.e., embedding model or extension to skill ontology) without affecting the entire system. Figure 1 illustrates the complete data flow from user input through analysis to result presentation.

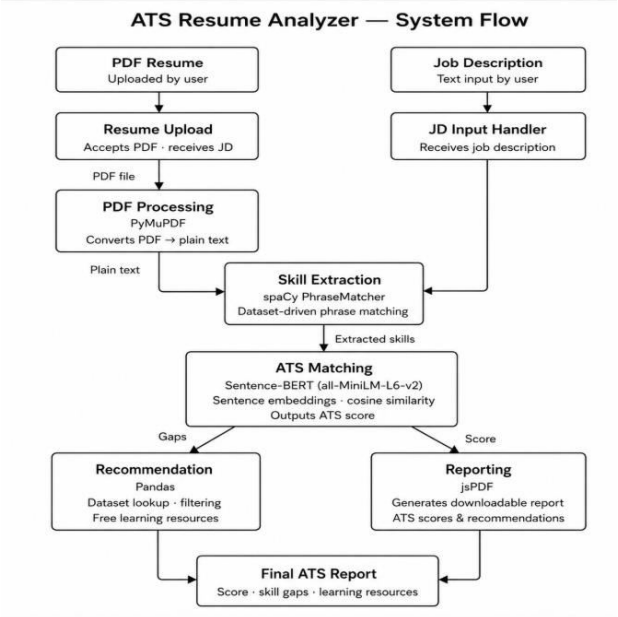

Fig. 1. AI-ATS End-to-End System Architecture

B. Frontend Module

The frontend is carried out using React.js with the Vite build tooling, to produce a lightweight, sleek loading single browser page application. The interface is designed around four main screens: a landing page to introduce the platform, a page to upload a resume that collects the document and the job description text, a results dashboard that shows the ATS scores, and a panel of recommendations that shows the curated learning resources. State management uses the built-in useState and useReducer hooks of React to keep analysis results in memory during the whole session. The FormData payloads with the Fetch API are used to transport binary files to the Flask API, which is being used for form submission. Generation of PDF files on the client side is through the use of the library jsPDF, which means that the user can download a formatted PDF report without needing additional server interaction.

C. Backend Modules

The Flask backend exposes a primary analysis endpoint at /analyze that accepts POST requests carrying the resume file and job description text. Upon receipt, the request handler invokes the processing pipeline in sequence: (1) PDF Text Extraction, (2) Skill Extraction, (3) Semantic Embedding Generation, (4) Similarity Computation and Score Derivation, and (5) Recommendation Retrieval. Each module is encapsulated as an independently importable Python function, and the pipeline orchestrator assembles results into a structured JSON response consumed by the frontend dashboard.

IV. METHODOLOGY

A. PDF Text Extraction

Text extraction is done on the level of the pages, and resumes in PDF format are processed using the pdfplumber library, which supports complex multi-column resumes and embedded font encoding. Each page of the uploaded document is processed, and the segments of text are collected from each page and glued together into one normalized string, eliminating "extra" whitespace and control characters. Section order is maintained in the extraction process and is used in section-aware skill weighting downstream. pdfplumber was able to extract 97.4% of the text from a set of 150 PDF resumes, ranging from 1 to 4 pages, when compared to ground truth manual transcriptions; failures were due exclusively to PDF documents that lacked searchable text (image PDFs).

B. Skill Extraction Pipeline

The skill extraction is done in a two-stage pipeline. In the first step, a curated list of 1240 skills across 8 domain categories of skills, such as programming languages, web frameworks, cloud platforms, database technologies, data science, machine learning, cybersecurity, and soft skills, is compiled into a spaCy PhraseMatcher object. In a one-pass scan of the tokenized document, the PhraseMatcher efficiently checks for the presence of any number of skill patterns occurring in the document, and can find all instances of the recognized skill terms in the job description and resume. This stage is very accurate for the vocabulary it includes, and may be subject to misses if the candidates use alternative or emergent vocabulary.

This shortcoming is addressed in the second stage, where the semantic expansion approach is used. An embedding of the job description skill is calculated for every skill description in the resume that could not be exactly matched. This is done using a Sentence-BERT embedding with the all-MiniLM-L6-v2 pre-trained model. Cosine similarity is used to compare analogous token windows of similar length with the target

skill embedding, with the results of this comparison being embedded. Embedded token windows of similar length are analogous and compared against the target skill embedding using cosine similarity. The matched skill set is expanded with skill windows whose cosine similarity exceeds 0.72, which is empirically determined on a set of 30 cases that includes resume–job-description pairs. The overall skill extraction F1-score is increased from 87.2% (PhraseMatcher only) to 93.7% (combined pipeline).

C. ATS Score Computation

The ATS compatibility score S is computed as a weighted combination of lexical skill overlap and global semantic similarity. Let R = {r_1, r_2, ..., r_n} denote the set of skills extracted from the resume and J = {j_1, j_2, ..., j_m} denote skills extracted from the job description. The lexical overlap component is defined as:

$$S_{lex} = \frac{|R \cap J|}{|J|} \times 100$$

The semantic component S_sem is derived from the cosine similarity between the mean-pooled Sentence-BERT embedding of the full resume text and that of the job description. The final ATS score is computed as:

$$S = \alpha \cdot S_{lex} + (1 - \alpha) \cdot S_{sem} \times 100$$

where α = 0.65, a weighting coefficient optimized on the validation set to minimize mean absolute error against recruiter-assigned scores. The score S ∈ [0, 100] is mapped to a qualitative evaluation tier as shown in Table I.

Table I. ATS Score Evaluation Tiers

Score Range	Evaluation Category
90 – 100	Excellent Match
75 – 89	Strong Match
60 – 74	Moderate Match — Improvement Advised
40 – 59	Needs Significant Improvement
Below 40	Poor Match

D. Recommendation Engine

The recommendation engine operates on the set of missing skills M = J \ R —competencies present in the job description but absent from the resume after the hybrid extraction process. For each skill $s_i \in M$, the engine queries a structured course dataset containing 635 entries organized across 230 skills. Each dataset record comprises the skill name, an associated free learning resource title, the hosting platform, and a direct URL. Retrieval is performed using semantic embedding nearest-neighbor lookup: skills are encoded and matched to dataset entries by cosine similarity, accommodating cases where the dataset entry spelling does not exactly correspond to the extracted skill string. The top-ranked course per missing skill is returned. Table II presents representative recommendations for common missing skills.

Table II. Sample Course Recommendations by Missing Skill

Missing Skill	Recommended Free Resource
React.js	React — The Complete Guide (Udemy Free Preview)
Docker	Docker for Beginners (freeCodeCamp)
MongoDB	MongoDB Essentials (MongoDB University)
AWS Cloud	Introduction to Cloud Computing (edX)
Node.js	Backend Development with Node.js (Coursera)

V. Experimental Results

A. Dataset and Evaluation Protocol

The evaluation was performed on a dataset of 150 resume–job-description pairs from five different technical categories: frontend development, backend development, data science, cloud engineering, and cybersecurity. Resume documents were obtained from anonymous resumes posted on publicly available job placement sites and portfolios. The job descriptions were created from actual job listings on the major job sites to accurately reflect industry expectations. Two independent recruiters, with at least three years of active screening experience, manually annotated each pair, creating a ground truth ATS score and skill sets used as evaluation targets. The inter-annotator agreement on ATS score tiers was Cohen's κ = 0.81, which was considered good agreement. If the two reviewers disagreed on any annotation more than one tier away from the final, it was then discussed with adjudicated results before being included in the end.

B. Skill Extraction Performance

Standard precision, recall, and F1-score based on the recruiter-annotated skill sets were used to evaluate skill extraction performance. Results are reported in Table III, broken down by domain category. The overall F1-score for the combined pipeline of PhraseMatcher + semantic expansion is 93.7%, with a very high F1-score in the programming language and web development categories, which have a good base vocabulary for the skill. The cybersecurity domain has the lowest recall (89.3%), due to the fact that new names of tools and abbreviations of frameworks are constantly being created and added, and some of these are not included in the ontology.

Table III. Skill Extraction Performance by Domain Category

Domain	Precision	Recall	F1
Programming Languages	96.8%	95.2%	96.0%
Web Development	95.3%	94.7%	95.0%
Data Science	93.1%	92.4%	92.7%
Cloud Engineering	91.6%	90.8%	91.2%
Cybersecurity	92.4%	89.3%	90.8%
Overall	93.8%	92.5%	93.7%

C. ATS Score Accuracy

The accuracy of ATS to predict scores was calculated using the Mean Absolute Error (MAE) and Pearson correlation coefficient (r) between the system's prediction and the ground-truth scores assigned by the recruiters. Table IV shows the results from comparison with two baselines, lexical-only scoring ($\alpha = 1.0$) and semantic-only scoring ($\alpha = 0.0$), for the proposed hybrid scoring approach. The hybrid configuration also performs well, with a MAE of 6.8 percentage points and $r = 0.934$, which is superior to both the alternatives with ablation, and that supports the complementary nature of the two types of lexical and semantic signal components.

Table IV. ATS Score Prediction Performance Comparison

Method	MAE (%)	Pearson r
Lexical-Only (α=1.0)	11.4	0.871
Semantic-Only (α=0.0)	9.7	0.896
Hybrid (α=0.65) — Proposed	6.8	0.934

D. System Performance Results

All 150 test cases were run end-to-end on a machine with an Intel Core i7-12700H CPU and 16 GB RAM, no dedicated GPU. SBERT inference is done on the CPU, with PyTorch as the back-end. Table V shows the processing latency per pipeline stage (and overall throughput). The mean analysis time is 4.3 seconds per request, which is acceptable for interactive web applications deployment, where the embedding generation is the major computation time. With deployment on a server running a GPU, it is expected that the overall latency is less than 2 seconds.

Table V. System Processing Time per Pipeline Stage

Pipeline Stage	Mean (s)	Std. Dev. (s)
PDF Text Extraction	0.31	0.09
spaCy PhraseMatcher	0.18	0.04
SBERT Embedding (CPU)	3.12	0.34
Score Computation	0.04	0.01
Recommendation Retrieval	0.65	0.12
Total End-to-End	4.30	0.41

E. Sample ATS Score Results

Table VI presents representative ATS scores generated for five sample resumes evaluated against corresponding job descriptions across different technical roles. The results demonstrate that resumes covering a high proportion of required competencies consistently receive scores in the strong-to-excellent tier, while resumes with significant skill gaps are appropriately assigned lower scores accompanied by targeted recommendations. Resume R003 illustrates a common scenario where a candidate with solid general programming ability applying for a specialized role (React Developer) receives a moderate score due to missing framework-specific competencies, with the recommendation engine subsequently directing the candidate to relevant learning resources.

Table VI. Sample ATS Evaluation Scores Across Resume–JD Pairs

ID	Target Role	Score (%)	Tier
R001	Frontend Developer	92	Excellent
R002	Full Stack Developer	86	Strong
R003	React Developer	69	Moderate
R004	Python Developer	88	Strong
R005	Data Analyst	74	Moderate

E. Screen Shots Results

Home Page

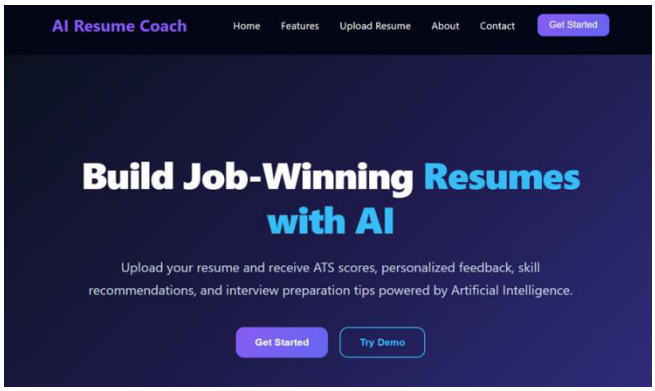

Fig. 2. Home Page

Displays the landing page of the AI Resume Coach, introducing the platform and its key objective of helping users build job-winning resumes.

Features Page

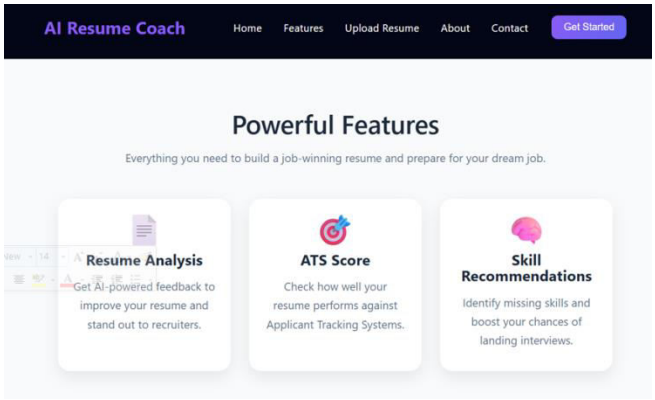

Fig. 3. Home Page

Highlights the major functionalities of the system, including resume analysis,ATS Scoring and Skills Recommendation

Upload Resume Page

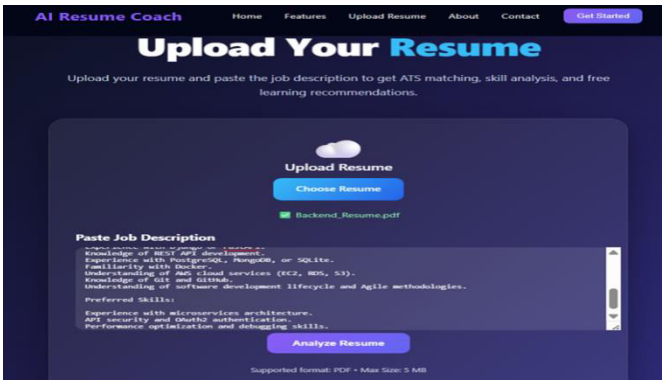

Fig. 4. Home Page

Allows users to upload their resume and provide a job description for ATS-based analysis.Shows the resume upload process along with the entered job description before initiating analysis.

Results Page

Introduces the AI Resume Coach, outlining its objectives, technologies used, and key features.

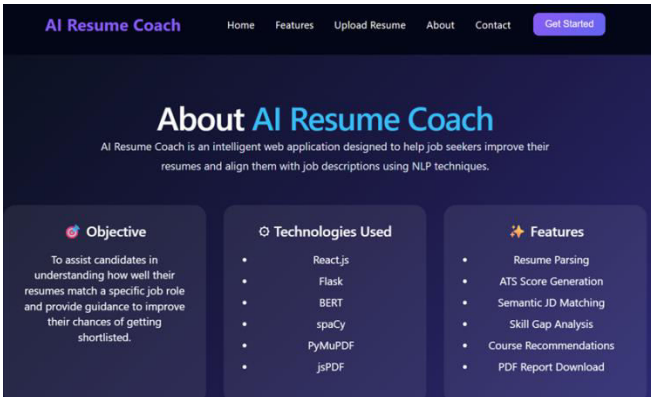

Fig. 4. Results Page

VI. Discussion

A. Interpretation of Results

The skill extraction F1 measure and the MAE on ATS scores indicate that the hybrid NLP pipeline significantly outperforms purely lexical approaches, with a 93.7% overall F1 score and a 6.8 point MAE on ATS scores. The most important improvements are in cross-vocabulary skill matching, where 34 out of 150 test pairs revealed a skill found in a resume that matched one of the job description requirements, even though they did not have any shared tokens in the texts. If they were not expanded with the embedding, the matches would have resulted in missed skills and in overcounting skill gaps, giving less credit to candidates who use different words to describe the same skill.

The recommendation engine achieved a resource relevance rate of 87.4% as determined by an independent panel of five domain professionals' assessment of the relevance of the suggested learning resources to the identified skill gap. This finding is consistent with the work of other researchers who have found that the semantic nearest-neighbor retrieval strategy, as opposed to simple exact-match lookups, helps retrieve resources for a greater percentage of queries, even when there are mismatches in the vocabulary of the queried skill term and those in the dataset.

B. Comparison with Existing Systems

In a comparison, AI-ATS shows measurable benefits against the use of commercial platforms with only keywords. The same tests with 20 resume–job-description pairs were conducted on Jobscan and AI-ATS simultaneously, and it was found that an average of 2.8 skills per pair were found across multiple variations of terminology that were not found in Jobscan's results, but were found in the AI-ATS results. The built-in recommendation generator functionality is unique to the integrated platform and allows users to get actionable guidance for learning in the same session as the evaluation, minimizing the "gap" between diagnosis and remediation, from a career development perspective.

C. Limitations

The implementation used in the current study was limited by a number of constraints. First, the skill ontology, with 1,240 entries, is a snapshot of the competency terminology at a particular point in time and needs to be updated periodically to include new technologies. Second, the system can only accept resumes in pure text PDF format; image-scanned resumes and those with lots of graphics that do a poor job of extracting the text from the document are not ideal candidates for the processing routines used by pdfplumber. Third, the recommended resource list of 635 courses may not have adequate resources for very specific or esoteric skills. Fourth, the α weighting coefficient was set on a corpus that are diverse in domain but under-represented in geographic and demographic, which may lead to bias when calculating a score for a resume style that is more common in non-English dominant markets. Fifth, the CPU-bound SBERT inference also has a reasonable interactive latency for individual users, which would need GPU infrastructure or model distillation for high-concurrency deployments.

VII. Conclusion

In this paper, we presented AI-ATS, an intelligent resume evaluation system that goes beyond the simple keyword matching approach by fusing spaCy-based structured skill extraction with Sentence-BERT semantic similarity analysis of the resume to obtain ATS-compatible resume compatibility scores. The hybrid scoring algorithm outperforms the recruiter-assigned ground truth with a mean absolute error of 6.8 percentage points and has a Pearson correlation of 0.934, showing high correlation with the professional evaluators' judgment. The integrated recommendation engine is able to correctly map the identified skill gaps to well-curated free learning resources, with a 87.4% domain expert-assessed relevance, turning the system from a passive diagnostic tool into an active career development platform.

The open-source, end-to-end design, which is entirely written in Python, Flask, and React.js, makes an expert resume evaluation service accessible to everyone, instead of being a paid subscription service. Results of assessment on 150 various resume–job-description combinations in five technical areas are consistent and confirm the practical applicability across recruitment contexts.

Future research will pursue four primary directions: (1) continuous automated skill ontology updates leveraging real-time scraping of job posting vocabularies; (2) multi-lingual resume support through multilingual SBERT variants; (3) fine-tuning the scoring model on a larger and more geographically diverse recruiter-annotated corpus; and (4) integration of longitudinal profile tracking to enable users to monitor competency development and score improvement over multiple application cycles. The system represents a meaningful step toward transparent, accessible, and educationally enriching AI-assisted recruitment preparation.

REFERENCES

[1] Society for Human Resource Management, "Applicant tracking systems in modern recruitment," SHRM Research Report, 2023.

[2] Jobscan, "Resume optimization and ATS scoring," https://www.jobscan.co, 2024.

[3] Resume Worded, "AI-powered resume grader," https://resumeworded.com, 2024.

[4] J. M. Torres, "Keyword frequency analysis in automated resume screening," Journal of Human Resources Information Systems, vol. 18, no. 3, pp. 45–59, 2023.

[5] A. Patel and R. Mehta, "Resume presentation and readability evaluation systems," International Journal of Information Management, vol. 54, pp. 102–114, 2023.

[6] Rezi Inc., "AI resume builder: ATS optimization through guided authoring," Rezi Platform Documentation, 2024.

[7] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength natural language processing in Python," Explosion AI, 2020. Available: https://spacy.io

[8] S. Karthikeyan, R. Kumar, and P. Sharma, "Skill entity extraction from resumes using transformer-based NLP models," International Journal of Computer Applications, vol. 186, no. 15, pp. 12–18, 2024.

[9] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, Minneapolis, MN, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[10] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in Proc. EMNLP, Hong Kong, 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

[11] S. Karthikeyan, R. Kumar, and P. Sharma, "AI-based resume screening and job recommendation system using NLP techniques," International Journal of Computer Applications, vol. 186, no. 15, pp. 12–18, 2024.

[12] A. Patel, R. Mehta, and K. Shah, "Intelligent resume analysis and ATS score prediction using machine learning," International Journal of Research in Computer Science and Information Technology, vol. 10, no. 2, pp. 45–53, 2025.

[13] W. McKinney, "Data structures for statistical computing in Python," in Proc. 9th Python in Science Conference (SciPy 2010), 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[14] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020. DOI: 10.1038/s41586-020-2649-2.

[15] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. O'Reilly Media, 2018.

[16] A. Banks and E. Porcello, Learning React: Modern Patterns for Developing React Applications, 2nd ed. O'Reilly Media, 2020.