IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

RESUME ANALYZER: AN INTELLIGENT RULE-BASED NLP SYSTEM FOR AUTOMATED RESUME EVALUATION AND ATS COMPLIANCE SCORING

[1]Vignesh R , [2]Mohamed Ali S,[3]Mohamed Arsath M, [4]Sriom A ,[5] Dr. R. Devi

[1234] UG Student, [5] Professor and Head

[12345]Department of Applied Computing And Emerging Technologies,

[12345] VELS Institute of Science, Technology and Advanced Studies (VISTAS),Chennai ,India

Abstract: The widespread adoption of Applicant Tracking Systems (ATS) has created a critical skill gap among job seekers, particularly fresh graduates who lack access to professional resume review tools. This paper presents the AI Resume Analyzer, a full-stack web application that performs multi-dimensional, rule-based Natural Language Processing (NLP) evaluation of resumes in PDF, DOCX, and TXT formats. The system calculates five distinct performance scores—ATS Compliance, Content Completeness, Format Quality, Keywords Coverage, and Writing Impact—yielding a weighted overall score. The backend is implemented in Python using the Flask framework, with text extraction handled by PyPDF2 and python-docx. A drag-and-drop HTML5 frontend renders animated score visualizations, section detection results, ATS issue diagnostics, and prioritized improvement suggestions. All analysis is executed in server memory with no persistent storage, ensuring complete user privacy. Testing across 15 real resume samples confirmed consistent accuracy and sub-3-second response times. The system demonstrates that lightweight, rule-based NLP achieves practical and transparent resume evaluation without requiring deep learning models or external APIs.

Keywords — Resume analysis, ATS optimization, NLP, Flask, Python, text extraction, information retrieval, career technology, keyword extraction, job market.

I. INTRODUCTION

The digital transformation of recruitment has fundamentally altered how organizations evaluate candidates. Applicant Tracking Systems (ATS) now serve as automated gatekeepers at most medium and large enterprises, parsing and ranking resumes before any human review occurs. Research indicates that more than 75% of resumes submitted to large corporations are filtered out by ATS systems before reaching a recruiter—not because candidates are unqualified, but because their resumes fail to meet automated structural and keyword requirements [1].

Traditional resume preparation resources, such as career counseling sessions, peer review, and online articles, provide general guidance but fail to address the specific technical requirements of ATS optimization. Commercial tools such as Jobscan and Resume.io offer automated scoring but are locked behind paid subscriptions, operate as black boxes without transparent scoring criteria, and require uploading sensitive personal data to remote servers.

This paper presents the AI Resume Analyzer, a locally-deployable, free, and transparent resume evaluation platform designed specifically for students and fresh graduates. The system applies a multi-stage rule-based NLP pipeline to evaluate resumes across five distinct performance dimensions and generate actionable, explainable feedback. The key contributions of this work are:

- A five-dimensional scoring framework covering ATS compliance, content completeness, format quality, keyword coverage, and writing impact.
- A fully transparent, rule-based NLP evaluation engine implemented in Python without dependency on commercial APIs or deep learning models.
- A privacy-first architecture that processes all resume content in server memory with zero persistent storage.
- An interactive HTML5 dashboard with animated score visualizations and color-coded diagnostic feedback.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the system architecture. Section IV details the NLP analysis methodology. Section V presents the scoring framework. Section VI discusses the experimental results. Section VII outlines future work, and Section VIII concludes the paper.

II. RELATED WORK

The field of automated resume analysis sits at the intersection of Natural Language Processing, Information Retrieval, and Human-Computer Interaction. Prior research has approached the problem through a spectrum of techniques ranging from rule-based systems to deep learning architectures.

Xu et al. [2] proposed a deep learning-based resume parsing framework using bidirectional Long Short-Term Memory (Bi-LSTM) networks for named entity recognition across resume documents. While effective for large-scale enterprise deployment, the approach requires substantial computational infrastructure and extensive labeled training data, making it unsuitable for lightweight local applications.

Luo et al. [3] introduced a hybrid model combining rule-based NLP with Support Vector Machine (SVM) classifiers for resume section detection. Their empirical validation confirmed that rule-based systems achieve competitive accuracy on structured documents such as resumes, a finding that directly motivates the design choice adopted in the AI Resume Analyzer.

Sinha et al. [4] conducted an empirical study on the quantifiable impact of ATS keyword optimization, finding that resumes tailored to match job description terminology received approximately 40% more recruiter callbacks compared to unoptimized equivalents. This result underlines the practical importance of automated ATS compliance checking.

Dardas et al. [5] proposed keyword optimization strategies specifically targeting automated resume screening, demonstrating that structured section detection combined with domain-specific keyword matching significantly improves ATS throughput rates. Bright et al. [6] further demonstrated that the presence of strong action verbs and quantified achievement statements measurably improves evaluator perception of resume quality, validating the writing impact scoring dimension in the proposed system.

Unlike prior work, the AI Resume Analyzer prioritizes accessibility, transparency, and privacy over accuracy at scale, targeting the underserved population of students and fresh graduates who require a free and locally-operable evaluation tool.

III. SYSTEM ARCHITECTURE

A. Architectural Overview

The AI Resume Analyzer follows a three-tier web application architecture comprising a browser-based frontend presentation layer, a Python Flask application server layer, and an in-memory analysis engine. The system operates in a fully stateless manner: each HTTP request initiates a complete analysis cycle, and all intermediate data objects are garbage-collected at the end of the request lifecycle, leaving no residual information in server memory between requests.

The application flow proceeds through four sequential stages: (1) file upload and validation, (2) format-specific text extraction, (3) multi-stage NLP analysis, and (4) score calculation and JSON response serialization. This pipeline is illustrated in Table I.

TABLE I

DATA FLOW ACROSS SYSTEM COMPONENTS

Source Component	Destination Component	Data Transferred
User Browser (File API)	Flask /analyze Endpoint	Binary file bytes (PDF/DOCX/TXT)
Flask Route Handler	Text Extraction Module	Raw file bytes + filename extension
Text Extraction Module	NLP Analysis Engine	Cleaned plain-text string
NLP Analysis Engine	Scoring Engine	Contact fields, section flags, skills list, verb count, metric count
Scoring Engine	JSON Response Builder	Five component scores + weighted overall score
Flask Response	Frontend Dashboard	Structured JSON analysis object

B. Backend Layer

The backend is implemented in Python 3.8+ using the Flask 2.x micro-framework. Two HTTP routes are defined: a GET / route that serves the frontend interface and a POST /analyze route that handles file upload and analysis. The MAX_CONTENT_LENGTH configuration parameter enforces a 10 MB file size limit as a denial-of-service mitigation measure. The application is designed for deployment on a local Flask development server or any WSGI-compatible production server such as Gunicorn.

C. Frontend Layer

The frontend is a single-file HTML5 application requiring no build tools or package managers. It uses the HTML5 File API for drag-and-drop resume upload, the Fetch API for asynchronous communication with the Flask backend, and vanilla JavaScript ES6+ for DOM manipulation and score visualization. Score rings are rendered using inline SVG with CSS stroke-dashoffset animation. The dashboard is fully responsive and supports desktop, tablet, and mobile viewports.

D. Software and Hardware Requirements

TABLE II

SOFTWARE REQUIREMENTS

Component	Specification
Operating System	Windows 10/11, Ubuntu 20.04+, or macOS 12+
Python Runtime	Python 3.8 or higher
Backend Framework	Flask 2.x
PDF Parsing Library	PyPDF2 3.x
DOCX Parsing Library	python-docx 0.8.x
Frontend Technologies	HTML5, CSS3, JavaScript ES6+
Minimum RAM	4 GB (8 GB recommended)
Storage	500 MB free disk space

IV. NLP ANALYSIS METHODOLOGY

A. Text Extraction

The text extraction module implements format-specific extraction algorithms for each of the three supported file types. For PDF files, PyPDF2.PdfReader iterates through all document pages, concatenating the text content extracted from each page. For DOCX files, the python-docx library iterates through all paragraph objects in the document body and joins their text content with newline delimiters. Plain TXT files are decoded directly from bytes using UTF-8 encoding with error substitution to handle non-standard characters. The module returns a single normalized text string that serves as the sole input to the downstream NLP pipeline.

B. Contact Information Extraction

Regular expression patterns are applied to the raw resume text to detect three critical contact information fields: email address, phone number, and LinkedIn profile URL. The email pattern matches standard RFC 5322 address formats. The phone pattern accepts international formats including country codes, parenthetical area codes, and various delimiter styles. The LinkedIn pattern matches canonical linkedin.com/in/ profile URLs. The presence or absence of each field contributes to the ATS compliance score as described in Section V.

C. Section Detection

A keyword dictionary maps each of six standard resume section names to a list of common textual variants. The detection algorithm converts the entire resume text to lowercase and performs substring matching for each section's keyword list. The six detected sections are Work Experience, Education, Skills, Projects, Professional Summary, and Certifications. A section is classified as present if any of its associated keywords appear anywhere in the lowercased resume text.

D. Skills Identification

Two curated skill lists are maintained as Python constants: a technical skills list of 14 domain-relevant technologies (python, java, javascript, react, sql, aws, docker, excel, power bi, machine learning, data science, duckdb, streamlit, git) and a soft skills list of 6 competencies (leadership, communication, teamwork, problem solving, management, agile). Skills detection uses substring matching against the lowercased resume text. The resulting list of detected skills is used both for display in the skills panel and as input to the Keywords Coverage scoring formula.

E. Writing Quality Analysis

Writing quality is evaluated through two distinct mechanisms. First, a list of 12 strong action verbs (achieved, built, created, designed, developed, improved, increased, led, managed, optimized, reduced, implemented) is checked against the resume text, and the count of matched verbs contributes to the Writing Impact score. Second, regular expressions identify numeric achievement patterns including percentage values (e.g., 45%), currency amounts (e.g., $10,000), and multiplier expressions (e.g., 3x). The count of such quantified achievements is incorporated into both the ATS Issues diagnostics and the Impact score calculation.

V. SCORING FRAMEWORK

The scoring engine calculates five distinct component scores, each targeting a different dimension of resume quality. A weighted combination of these scores produces the final overall score. Table III summarizes the scoring dimensions and their respective weights.

TABLE III

SCORE COMPONENT DEFINITIONS AND WEIGHTS

Score Component	Weight	Calculation Basis
ATS Compliance Score	30%	Contact completeness, section presence, word count adequacy, quantification density
Content Completeness Score	20%	Section count ratio, word count, quantification density
Format Quality Score	15%	ATS score threshold, contact completeness, word count range
Keywords Coverage Score	15%	Detected skills count, weighted at 10 points per detected skill
Writing Impact Score	20%	Action verb count (x8) plus quantified achievement count (x10)

The ATS Compliance Score begins at 100 and applies deductions for each detected deficiency: 15 points for a missing email address, 15 points for a missing phone number, 10 points for the absence of a professional summary section, 15 points for a word count below 150, and 10 points for fewer than two quantified achievements. The score is floored at 20 to preserve the incentive structure. The final overall score is computed as:

$$S_{overall} = 0.30 \cdot S_{ATS} + 0.20 \cdot S_{Content} + 0.15 \cdot S_{Format} + 0.15 \cdot S_{Keywords} + 0.20 \cdot S_{Impact}$$

Score thresholds for user-facing performance categories are defined as: Excellent (≥80), Good (60–79), Fair (40–59), and Needs Work (<40). These thresholds are displayed with color-coded indicators in the frontend dashboard (green, blue, yellow, and red respectively).

VI. EXPERIMENTAL RESULTS

A. Test Methodology

The AI Resume Analyzer was evaluated through six testing categories: unit testing of individual analysis functions with controlled text inputs, integration testing of the complete HTTP request-response cycle using the Flask test client, system testing across a diverse set of 15 real resume samples, user acceptance testing with students and placement staff, negative testing with invalid and malformed inputs, and performance testing under concurrent upload conditions.

B. Functional Test Results

TABLE IV

SUMMARY OF FUNCTIONAL TEST CASES

Test Case ID	Objective	Expected Output	Result
TC-01	PDF upload and text extraction	All five scores returned	PASS
TC-02	DOCX upload and section detection	5 sections detected correctly	PASS
TC-03	ATS score deduction for missing	ATS score reduced by 30 points	PASS

	contact fields		
TC-04	Technical skills detection	4 matching skills returned	PASS
TC-05	Rejection of unsupported file format (.jpg)	HTTP 400 with error message	PASS
TC-06	Detection of all 6 standard resume sections	sections_missing = []	PASS
TC-07	Action verb count and impact score calculation	verb_count = 5, impact_score = 70	PASS
TC-08	Graceful handling of empty file	HTTP 400 with descriptive error	PASS

C. Performance Results

All functional test cases passed successfully. System testing across 15 real resume samples confirmed that the analysis engine produces consistent and reliable scores across varying resume quality levels and file formats. End-to-end analysis time averaged 1.4 seconds for typical one-to-two-page PDF resumes on standard consumer hardware (Intel Core i5, 8 GB RAM), and remained below 3 seconds for all tested samples. No memory leaks were observed across 100 sequential analysis requests, confirming the effectiveness of the stateless, in-memory architecture.

D. Comparison with Existing Tools

TABLE V

FEATURE COMPARISON: AI RESUME ANALYZER VS. COMMERCIAL TOOLS

Feature	**AI Resume Analyzer**	**Commercial Tools (e.g., Jobscan)**
Cost	100% Free	Paid ($15–$40/month)
Privacy	Local processing only	Cloud upload required
Scoring Transparency	Full criteria explained	Black-box scoring
Offline Operation	Fully offline capable	Requires internet
Section-Level Diagnostics	Yes — per section detail	Generic percentage only
Action Verb Detection	Yes — 12 verb patterns	Not available
Quantification Check	Yes — regex-based	Not available
Student-Focused Design	Tailored for fresh graduates	Generic audience

VII. FUTURE WORK

Several enhancements are planned for subsequent versions of the AI Resume Analyzer. The most impactful improvement would be integration of a Large Language Model (LLM) backend to enable semantic evaluation of writing quality, contextual achievement assessment, and fluent natural language improvement suggestions. Initial experiments with open-source models such as Llama 3 and Mistral confirm feasibility on consumer hardware.

A sophisticated job description matching engine is under development that would calculate keyword overlap scores between a candidate's resume and a specific job posting, provide role-specific tailoring recommendations, and identify critical missing keywords. Additional planned enhancements include: user account management with resume version history and improvement tracking; a real-time resume editing environment with live score updates; native mobile applications for iOS and Android; cloud deployment with a documented REST API for integration into university placement portals; advanced domain-specific skill taxonomies for Data Science, Web Development, Cybersecurity, and other specializations; and multi-language support covering Tamil, Hindi, French, German, and Spanish resume formats.

VIII. CONCLUSION

This paper presented the AI Resume Analyzer, a full-stack web application that transforms the traditionally subjective and opaque process of resume evaluation into a systematic, data-driven, and transparent assessment workflow. The system applies a multi-stage rule-based NLP pipeline to evaluate resumes across five dimensions—ATS compliance, content completeness, format quality, keyword coverage, and writing impact—generating a weighted overall score and a prioritized set of actionable improvement recommendations.

The system's principal technical contributions are its five-dimensional scoring framework, its privacy-first in-memory processing architecture, and its commitment to transparent, explainable feedback. Unlike commercial tools that provide opaque scores behind subscription paywalls, the AI Resume Analyzer exposes every scoring criterion and deduction rule, educating users about ATS optimization principles rather than simply rating their documents.

Experimental evaluation across 15 real resume samples confirmed consistent analysis accuracy and sub-3-second response times on standard consumer hardware. All functional test cases passed successfully, including edge cases involving invalid file formats, empty files, and resumes with missing critical sections. User acceptance testing with students and placement staff validated that the system's feedback is practical, understandable, and directly actionable.

The AI Resume Analyzer demonstrates that lightweight, rule-based NLP systems can deliver significant practical value in career development technology without requiring deep learning infrastructure, cloud services, or paid APIs. The system is openly deployable and directly addresses the needs of the large population of students and fresh graduates who currently lack access to professional resume optimization guidance.

REFERENCES

[1] L. Xu, F. Luo, Y. Geng, and H. Li, "Resume Parsing Using Deep Learning and Named Entity Recognition," in Proc. ACM Int. Conf. Inf. Knowl. Manag. (CIKM), 2018, pp. 2353-2356.

[2] X. Luo, D. Ye, R. Pan, and J. Lu, "Hiring Now: A Resume-Based Approach to Automated Candidate Screening," IEEE Trans. Knowl. Data Eng., vol. 31, no. 5, pp. 977-991, May 2019.

[3] R. Sinha, A. Patel, and S. Kumar, "Impact of ATS Optimization on Recruiter Response Rates: An Empirical Study," J. Human Resource Manag., vol. 14, no. 3, pp. 112-128, 2021.

[4] N. Dardas, M. Al-Hamami, and A. Obeidat, "Keyword Optimization Strategies for Automated Resume Screening Systems," Int. J. Inf. Manag., vol. 58, pp. 102-118, 2022.

[5] J. Bright, D. Earl, J. Adams, and R. Morrill, "The Effect of Action Verb Usage and Quantified Achievement Statements on Resume Evaluation," J. Appl. Commun. Res., vol. 48, no. 2, pp. 231-248, 2020.

[6] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Hoboken, NJ: Pearson, 2021.

[7] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA: O'Reilly Media, 2019.

[8] Flask Documentation, Flask Web Development Framework, Pallets Projects, 2024. [Online]. Available: https://flask.palletsprojects.com/

[9] PyPDF2 Documentation, PDF Parsing with Python, 2024. [Online]. Available: https://pypdf2.readthedocs.io/

[10] python-docx Documentation, Working with Microsoft Word Documents in Python, 2024. [Online]. Available: https://python-docx.readthedocs.io/