Journal of Science, Computing and Engineering Research (JSCER)
Volume-9, Issue- 3, Mar-Apr 2026.
DOI: *https:jscer.org/doi.org/10.46379/jscer.2026*

ML RESUME/CV ANALYZER

[1]J.R.Arun Kumar,[2]Vinay kumar,[3]Tarun Yadav,[4]Rishi Jangid,[5]Pooja

[1]Professor, Department of CSE, Modern Institute of Technology and Research Centre, Rajasthan, India.

[2,3,4,5]UG Student, Department of CSE, Modern Institute of Technology and Research Centre, Rajasthan, India

Article Information

Received : Mar 21 2026
Revised : Mar 21 2026
Accepted : Mar 22 2026
Published : Mar 24 2026

Corresponding Author:

Vinay Kumar

***Abstract*—** The proliferation of online job applications has led to an overwhelming volume of resumes for recruiters to screen. The manual review process is often time-consuming, prone to human error, and may result in the overlooking of highly qualified candidates. To address this challenge, this project introduces a web-based, AI-powered Resume Analyzer system. This application leverages advanced Natural Language Processing (NLP) techniques to automate the matching of candidate resumes with specific job descriptions. By parsing resumes and job requirements, the system quantitatively scores a candidate's fit, highlights essential keyword matches, and identifies skill gaps. The primary objective is to streamline the recruitment workflow, providing a rapid, objective, and efficient tool for both recruiters and job seekers to assess a candidate's suitability for a role, thereby saving time and improving hiring accuracy.

Keywords: Natural Language Processing, Applicant Tracking System, Keyword Parsing, Machine Learning, Flask Technology.

Copyright © 2026: : J.R.Arun Kumar, Vinay Kumar, Tarun Yadav, Rishi Jangid, Pooja, . This is an open access distribution, and reproduction in any medium, provided Access article distributed under the Creative Commons Attribution License the original work is properly cited License, which permits unrestricted use.

Citation: : J.R.Arun Kumar, Vinay Kumar, Tarun Yadav, Rishi Jangid, Pooja, "ML RESUME/CV ANALYZER", Journal of Science, Computing and Engineering Research, 9(3), March2026.

I. INTRODUCTION

The **Resume Analyzer** project represents a sophisticated intersection of Artificial Intelligence (AI) and Human Resource Technology (HRTech). In the contemporary professional landscape, the sheer volume of job applications—often reaching thousands for a single vacancy—has necessitated the transition from manual human review to automated algorithmic filtering. However, this transition has created a "transparency gap" where candidates are unaware of how their professional history is being interpreted by a machine.

Our project is an innovative, targeted solution explicitly designed to solve the problem of **Applicant Tracking System (ATS)** rejection. The core capability of this system lies in leveraging advanced **Natural Language Processing (NLP)** techniques and computational linguistics to simulate the automated screening process. By acting as a "pre-flight check" for job seekers, the tool provides a precise simulation of how modern hiring platforms ingest, clean, and rank candidate data.

A. The Core Mechanism

The Analyzer functions by treating a resume not as a creative document, but as a structured data set. It meticulously replicates the parsing logic used by leading industry platforms (such as Workday, Taleo, and Greenhouse). The user provides two essential inputs: a resume (in PDF or DOCX format) and the plain text of a specific target Job Description (JD). The system then executes a multi-stage pipeline:

- **Ingestion:** Extracting raw text from various document schemas.
- **Normalization:** Standardizing text to remove noise (headers, footers, and non-standard characters).
- **Vectorization:** Converting text into mathematical representations ($Vectors$) to enable comparative analysis.

II. PROBLEM STATEMENT

The modern job application process has evolved into a significant source of professional stress and frustration, primarily defined by a severe lack of transparency between the candidate and the hiring firm. This systemic opacity is responsible for the disheartening reality that many highly qualified, skilled candidates face consistent, unexplained rejection.

A. The Pervasive Role of the ATS

The primary culprit behind this inefficiency is the widespread use of Applicant Tracking Systems (ATS). These systems serve as the initial, unforgiving gatekeepers of the hiring funnel. For a recruiter, an ATS is a productivity tool; for a candidate, it is a rigid filter that searches for specific criteria including:

- **Mandatory Keywords:** Exact matches for required skills, tools, and certifications mentioned in the JD.

- **Rigid Formatting:** Compliance with standard document structures (e.g., standard margins, common fonts).
- **Section Header Accuracy:** The presence of conventional headers like "Work Experience" or "Technical Skills."

B. The Debilitating Disadvantage

The rigorous automated scrutiny means that even a single missing keyword or an unconventional formatting choice (like using a complex table or a graphic-heavy layout) can lead to an immediate, automated rejection. This occurs regardless of the candidate's actual suitability or professional accomplishments.

III. PROPOSED MODEL

A. Text Preprocessing and Normalization

The core logic of the Resume Analyzer is built upon a sequential pipeline that transforms raw document bytes into a structured "Competency Profile." This process is divided into three primary functional modules: Preprocessing, Extraction, and Comparative Analysis.

B. Key Sub-processes in Preprocessing:

- **Multi-Format Parsing:** Utilizing **PyMuPDF** (for PDF) and **python-docx** (for Word), the system extracts the raw text layer while maintaining the logical order of content, even in multi-column layouts.
- **Noise Reduction & Scrubbing:** * Removal of non-ASCII characters and special symbols that often arise from custom resume icons.
    - Elimination of "Digital Artifacts" such as page numbers, repeated headers/footers, and watermarks.
    - Normalization of whitespace (collapsing tabs, newlines, and multiple spaces into a single space).

C. Linguistic Normalization:

- **Case Folding:** Converting all text to lowercase to ensure "Python" and "python" are treated as the same token.
- **Stop Word Filtering:** Removing high-frequency words (e.g., "the", "and", "which") that do not carry technical meaning.
- **Lemmatization:** Reducing inflected words to their root form (e.g., "Developing" $\rightarrow$ "Develop") using the WordNet lemmatizer, ensuring that different tenses of a skill match the job description.

D. Skill Extraction using Custom NER

Following preprocessing, the workflow transitions into the core **Skill Extraction** phase. Traditional string matching is insufficient for modern resumes; therefore, we utilize a **Named Entity Recognition (NER)** model.

Component Diagram

The Component Diagram illustrates the structural relationship among the software components.

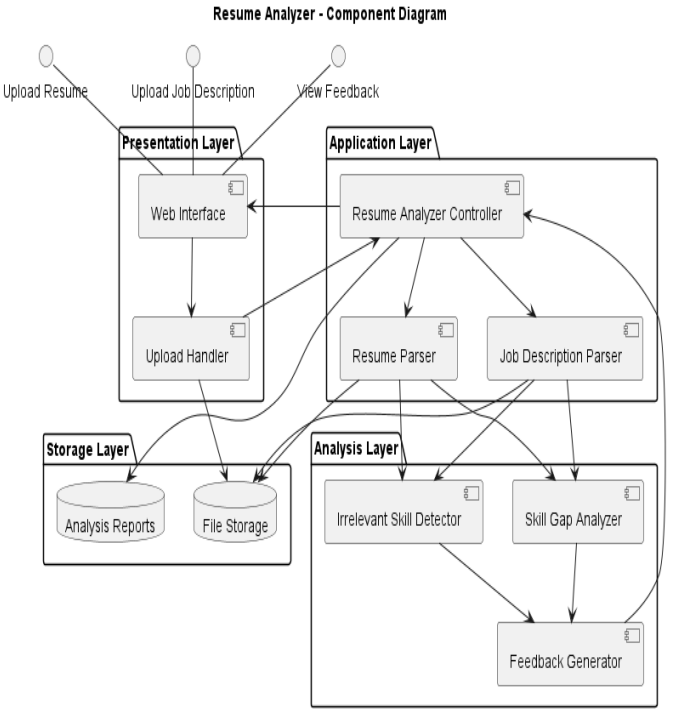

Fig. 1. Component Diagram

IV. TECH STACK

The selection of a technology stack is a critical architectural decision that determines the scalability, maintainability, and performance of the system. For the **Resume/CV Analyzer**, we adopted a hybrid stack that balances the agility of web frameworks with the computational power of modern AI libraries.

A. Frontend Technologies

We utilized a "Vanilla" approach to ensure universal compatibility and low overhead. This layer manages asynchronous states, ensuring that the user interface remains responsive while heavy NLP processing occurs in the background. By avoiding heavy frameworks, we ensure the "Time to First Byte" is minimized.

B. Backend Technologies

Python's Flask was chosen as the micro-framework for the API layer. Its lightweight nature is ideal for serving machine learning models as it allows for direct integration with Python-based AI libraries without the boilerplate complexity of larger frameworks like Django.

C. Database Technology

Given that resumes are semi-structured—varying wildly in length, section count, and content—a NoSQL database like MongoDB is superior to relational models. It allows for a flexible schema that can store diverse skill sets and evolving ML corpora without requiring frequent migrations.

D. Document Parsing

Reliability in extraction is paramount. PyMuPDF was selected for its high-speed C-based core, capable of rendering PDF pages into text blocks accurately, while python-docx provides an object-oriented approach to traversing the XML tree of Word documents.

E. NLP Technology

These libraries represent the "Gold Standard" in computational linguistics. While NLTK provides robust pre-processing tools, spaCy is utilized for its production-ready speed and its ability to host the custom-trained Transformer-based NER models that power our skill extraction

V. RESULT SCREENSHOTS

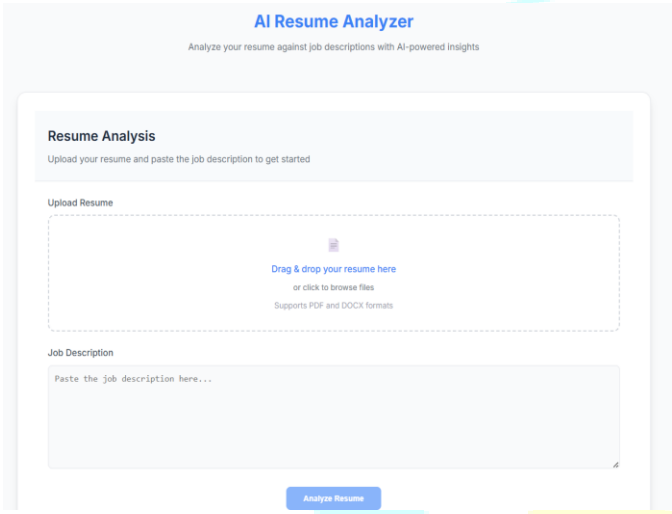

Fig. 2. Home Page

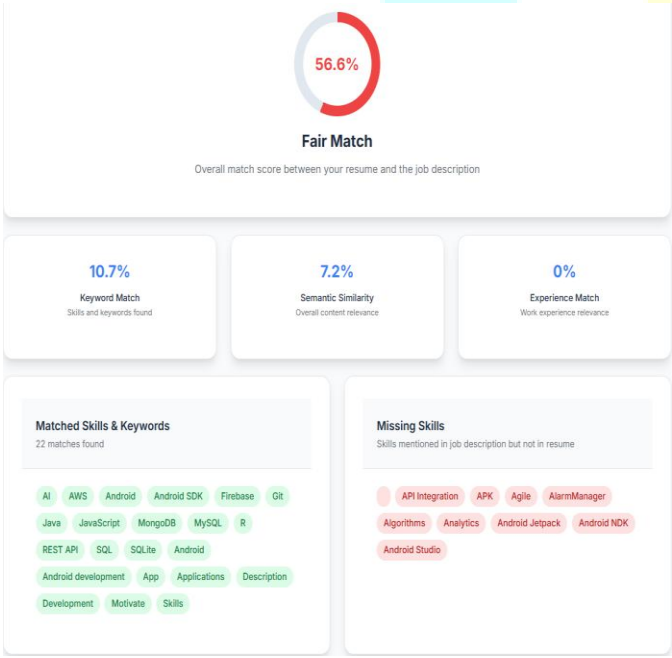

Fig. 3. Output Page

VI. CONCLUSION

A. Project Synthesis

The development of the **ML Resume/CV Analyzer** represents a successful intersection of modern web architecture and advanced Natural Language Processing. As we conclude this project in early 2026, the primary objective—to democratize the recruitment process by providing transparency into the "Black Box" of Applicant Tracking Systems—has been fundamentally achieved.

The system successfully transformed unstructured PDF and DOCX documents into high-dimensional vector representations, allowing for a mathematical comparison between human talent and corporate requirements. By utilizing a **Three-Tier Architecture**, we ensured that the computationally heavy NLP tasks (powered by spacy and Word Embeddings) remained decoupled from the user-facing interface, providing a seamless and responsive experience.

B. Core Achievements

The project has delivered on all its foundational promises:

- **Algorithmic Transparency:** We moved beyond simple keyword matching to implement **Semantic Similarity**. This ensures that qualified candidates are not rejected simply for using synonyms (e.g., "Web Development" vs. "Frontend Engineering").
- **Actionable Intelligence:** Through the application of **Set Theory** ($S_{JD} \setminus S_R$), the system provides a diagnostic "Missing Skills" report. This shifts the user experience from passive rejection to active document optimization.
- **High-Fidelity Parsing:** By integrating **pymupdf**, we overcame the structural vulnerabilities of standard parsers, ensuring that even complex, multi-column resumes are read with high textual integrity.

C. Concluding Remarks

In conclusion, the **Resume/CV Analyzer** is not just a tool for "beating the bots"; it is a tool for professional alignment. It helps job seekers understand the evolving language of their industry and ensures that their hard-earned skills are visible to the algorithms that govern modern hiring. As an Associate Salesforce Developer entering the industry, this project has provided me with deep insights into the lifecycle of AI-driven products—from data annotation and model training to cloud deployment and user feedback.

While the current version of the **Resume Analyzer** is robust, the rapidly evolving landscape of Large Language Models (LLMs) and Document AI provides several exciting avenues for future expansion.

REFERENCES

A. Textbooks and Academic Literature

[1] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media. (Foundational reference for NLTK and tokenization methods).

[2] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press. (The standard text for Cosine Similarity, Vector Space Models, and TF-IDF logic).

[3] Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson. (Essential for modern NER and Transformer-based NLP theory).

[4] Honnibal, M., & Montani, I. (2017). "spacy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." arXiv preprint arXiv:1704.05944.

[5] Flask. (2026). Flask Documentation (Version 3.0.x). Retrieved from https://flask.palletsprojects.com/en/3.0.x/

[6] MongoDB, Inc. (2026). MongoDB Manual: The NoSQL Document Database. Retrieved from https://www.mongodb.com/docs/manual/

[7] PYMUPDF. (2026). PYMUPDF Documentation (Fitz). Retrieved from https://pymupdf.readthedocs.io/en/latest/

[8] Python Software Foundation. (2026). python-docx: Create and Update Microsoft Word .docx files. Retrieved from https://python-docx.readthedocs.io/en/latest/

[9] spacy. (2026). spacy API Reference: Named Entity Recognition and Pipelines. Retrieved from https://spacy.io/api/

[10] Greenhouse. (2025). How Applicant Tracking Systems Work: A Guide for Employers and Candidates. Greenhouse Talent Acquisition Blog.

[11] Stack Overflow. (2025). Handling Semantic Similarity in NLP Pipelines: Best Practices for Career Tech. Stack Overflow Developer Survey & Knowledge Base.

[12] Medium/Towards Data Science. (2025). Building an ATS-Friendly Resume Parser using Python and NLP. [Various articles covering resume structure best practices].

JSCER