

AI Resume Analyzer using Generative AI

Shrutika Dadasaheb Newase

Department of Computer Engineering

Dattakala Group of Institutions, Faculty of Engineering, Pune, Maharashtra, India

Guide: Mrs. Vhandkar P.S.

Email: newaseshrutika@gmail.com | Mobile: 9325825112

Abstract

Recruitment teams and job seekers often face difficulty matching resumes with job descriptions because resumes use varied formats, terminology, and descriptions of skills. This paper proposes an AI Resume Analyzer using Generative AI to extract resume information, compare candidate profiles with job requirements, identify relevant skills, and generate understandable feedback. The proposed workflow combines document parsing, text preprocessing, skill and experience extraction, semantic similarity, and a generative model that produces a structured analysis. The system is intended to support—not replace—human decision-making by explaining matches, highlighting missing or unclear information, and suggesting resume improvements. A prototype may use PDF/DOCX parsing, embeddings, a vector similarity method, and a controlled language-model prompt. Evaluation should examine extraction quality, relevance of recommendations, consistency, latency, and fairness across varied resume formats. No experimental performance results are claimed because the proposed system has not been benchmarked. The design emphasises consent, privacy, transparency, and avoidance of unsupported or discriminatory hiring conclusions.

Keywords: Generative AI, Resume Analysis, Natural Language Processing, Semantic Matching, Recruitment, Large Language Models

1. Introduction

Resumes are a common way for applicants to communicate education, experience, projects, and skills. However, recruiters may receive many applications for a single role, while applicants may find it difficult to understand whether their resumes clearly demonstrate the capabilities listed in a job description. Simple keyword matching can miss equivalent terms and may rank documents poorly when the wording differs even though the underlying skills are similar.

Generative AI and natural language processing can help transform unstructured resume text into structured information and provide contextual explanations. Yet language models can make unsupported claims, overlook details, or reproduce bias present in data. A responsible resume analyzer should therefore ground its output in the submitted documents, show evidence for each match, and keep final decisions with a qualified human reviewer.

1.1 Problem Statement

The problem is to analyse resumes and job descriptions consistently, identify evidence-backed skill and experience matches, and provide useful feedback while protecting applicant data and avoiding misleading or discriminatory recommendations.

1.2 Objectives

- Extract key resume sections such as education, skills, experience, and projects.
- Compare resume content with the requirements of a selected job description.
- Use semantic matching to recognise related skills beyond exact keyword overlap.
- Generate clear, evidence-based feedback and improvement suggestions.
- Evaluate output quality, privacy, fairness, and reliability.

1.3 Scope

The system is intended as an academic decision-support prototype for resume review and self-improvement. It does not make hiring decisions, verify credentials, or guarantee employment outcomes.

2. Literature Review

Resume screening has traditionally used keyword search, rule-based filters, and structured applicant-tracking systems. These approaches are relatively easy to interpret but may fail to recognise synonyms, context, or transferable skills. Natural language processing methods support text classification, named-entity extraction, and semantic representation, which can improve the handling of unstructured documents. Sentence embeddings and similarity measures can estimate the relationship between resume content and job requirements without requiring exact wording.

Large language models introduce the ability to summarise experience and generate natural-language feedback. However, generated output may include unsupported details, inconsistent ratings, or bias. Consequently, a practical system should constrain prompts, preserve links to source text, and evaluate outputs against defined criteria rather than treating model-generated scores as objective truth.

2.1 Research Gap

A resume analyzer needs to combine semantic matching with transparent evidence and privacy-aware processing. A high similarity score alone cannot prove that a candidate is qualified. The proposed design addresses this gap by separating extraction, matching, and feedback generation, while making limitations visible to the user.

3. Proposed System

The proposed system accepts a resume file and a job description. It extracts text, identifies relevant sections, normalises skill terms, and builds semantic representations of resume statements and job requirements. A matching component estimates alignment and records supporting text. A generative AI component then converts these findings into a structured report containing matched skills, potentially missing requirements, unclear statements, and suggestions for improving clarity. The user can review and edit the report before using it.

Figure 1. Proposed processing flow (conceptual):

Resume + Job Description → Text Extraction → Preprocessing → Skill/Experience Extraction → Semantic Matching → Evidence-Grounded Generative Feedback → Reviewable Report

3.1 Main Modules

Module	Purpose
Document ingestion	Accept resume files and job descriptions with file-type and size validation.
Text extraction	Read text from PDF/DOCX documents and identify extraction failures.
Preprocessing	Clean text, split sections, normalise terms, and preserve source references.
Matching engine	Compare skills, experience, education, and job requirements using rules and semantic similarity.
Generative feedback	Produce structured explanations and suggestions grounded in extracted evidence.
Report and privacy	Display findings, support user review, and apply controlled storage and deletion.

3.2 System Architecture

A prototype may use Python for document processing, a PDF/DOCX parsing library for extraction, an embedding model for semantic representation, and a generative AI API or locally hosted language model for feedback. FastAPI can expose authenticated endpoints, while SQLite or PostgreSQL can store minimal report metadata. A web interface can allow users to upload a resume, paste a job description, inspect evidence, and download the report. Model and provider choices should be documented for reproducibility.

4. Methodology

4.1 Data and Consent

The prototype can be evaluated using consented sample resumes, synthetic examples, or appropriately licensed public data. Personal contact details should be removed or masked where they are not required. Uploaded resumes should not be used for model training without explicit permission, and retention periods should be clearly communicated.

4.2 Preprocessing

- Validate file type and size before parsing.
- Extract text and detect scanned documents that may require OCR.
- Separate resume sections such as skills, education, projects, and experience.
- Normalise common skill names while preserving the original wording.
- Record evidence snippets so that each reported match can be checked.

4.3 Matching and Generative Model

The system can combine exact skill matching with embedding-based semantic similarity. Job requirements should be represented as individual criteria rather than a single paragraph. Each criterion can be associated with a supporting resume excerpt or marked as not found in the supplied document. The language model should receive only the relevant extracted content and be instructed not to invent qualifications, infer protected characteristics, or treat missing information as proof that a person lacks a skill.

4.4 Report Generation

The report may include a summary of relevant experience, matched requirements with evidence, requirements that are not clearly demonstrated, resume clarity suggestions, and questions for the applicant to verify. If a numerical match indicator is included, its meaning and limitations should be explained; it should not be presented as a probability of job success.

4.5 Conceptual Algorithm

1. Receive the resume and job description with user consent. 2. Validate files and extract text. 3. Segment resume and job requirements. 4. Extract and normalise skills and experience statements. 5. Calculate exact and semantic matches. 6. Attach evidence snippets to each finding. 7. Generate feedback using constrained prompts. 8. Validate the output and display it for human review.

5. Implementation Plan

Layer	Suggested technology	Responsibility
Document processing	Python, PDF/DOCX parser, OCR if needed	Extract and clean resume text.
NLP and matching	Embeddings, similarity measures, skill dictionary	Map resume evidence to job criteria.
Generative AI	LLM API or local model	Produce evidence-grounded feedback.
API	FastAPI	Handle uploads and analysis requests securely.
Storage	SQLite prototype; PostgreSQL extension	Store minimal report and configuration metadata.
Interface	React or HTML/CSS/JavaScript	Show matches, evidence, and suggestions.

5.1 Security and Privacy Requirements

- Use authenticated access and restrict who can view uploaded documents.
- Encrypt data in transit and protect stored files.
- Keep API keys outside source code and use environment variables or a secret manager.
- Minimise personal data sent to external model providers.
- Provide clear retention and deletion controls.
- Log system errors without unnecessarily recording resume contents.

6. Evaluation Plan and Discussion

The prototype should be evaluated using varied resume formats and job descriptions. Human reviewers can assess whether extracted fields are correct, whether matches are supported by evidence, and whether generated suggestions are relevant and actionable. Testing should include ambiguous wording, missing sections, different document layouts, and cases where the model might otherwise generate unsupported claims.

Metric	Meaning
Extraction precision/recall	Correctness and completeness of extracted fields or skills.
Match relevance	Whether identified matches accurately reflect the job criteria and resume evidence.
Feedback quality	Human-rated usefulness, clarity, and actionability of suggestions.
Unsupported-claim rate	Frequency of generated statements not supported by the source documents.
Latency	Time from submission to report generation.
Fairness review	Check for inappropriate differences or use of protected attributes.

No numerical results are reported because the proposed system has not yet been tested in a controlled experiment. A completed evaluation should document the test samples, annotation procedure, model version, prompts, thresholds, reviewer agreement, and known limitations.

7. Advantages and Limitations

7.1 Advantages

- Recognises related terminology through semantic matching.
- Provides structured, readable feedback for applicants and reviewers.
- Links findings to evidence in the source resume.
- Can be extended to different roles and resume formats.

7.2 Limitations

- PDF parsing and OCR may fail on complex layouts or scanned files.
- Similarity scores can overstate or understate the actual relevance of a skill.
- Generative models can produce inaccurate or inconsistent feedback.
- Performance depends on model choice, prompts, language, and document quality.
- The system cannot independently verify the truth of a candidate's claims.

8. Ethical and Responsible Use

The system should be used as an assistive tool rather than an automated hiring authority. It should not infer sensitive personal characteristics or rank applicants based on protected attributes. Users should know how their information is processed, and applicants should be able to review and correct extracted information. Human reviewers must consider context and avoid treating a missing resume keyword as proof of missing ability.

9. Future Scope

Future work may include multilingual resume support, improved OCR, role-specific skill taxonomies, explainability views, local/private model deployment, integrations with applicant-tracking systems, and a larger evaluation with independent human reviewers. Further work should test reliability across industries, experience levels, and document formats.

10. Conclusion

This paper proposed an AI Resume Analyzer using Generative AI to extract resume information, compare it with job requirements, and generate evidence-grounded improvement suggestions. The design combines document parsing, semantic matching, and constrained language-model feedback while emphasising privacy, transparency, fairness, and human review. Its practical usefulness must be established through systematic testing, and generated analysis should support—not replace—human judgement.

References

- [1] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*, 4171–4186.
- [2] Reimers, N., and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of EMNLP-IJCNLP*, 3982–3992.
- [3] Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.
- [4] National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1.
- [5] Raghavan, M., Barocas, S., Kleinberg, J., and Levy, K. (2020). Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices. *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, 469–481.