IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

AI-Based Resume Filter System: Intelligent Candidate Screening and Job Matching

Nooka Harshavardhan[1], T. V. Sai Prudhvi[2], R. Bharath[3]

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[1-3]

Abstract: Recruitment teams often receive a large number of resumes for a single job opening. Manually reviewing every application can consume significant time and may lead to inconsistent shortlisting. This paper proposes an AI-based Resume Filter System that assists recruiters by extracting information from resumes, identifying relevant skills, comparing candidate profiles with a job description, and producing an explainable shortlist for human review. The proposed workflow accepts resumes in PDF or DOCX format, extracts and preprocesses text, identifies education, experience, skills, and relevant keywords, and represents candidate information for comparison with job requirements. A matching and ranking component can combine skill coverage, relevant experience, education requirements, and semantic similarity to generate a candidate relevance score. The system is intended as a decision-support tool rather than an autonomous hiring authority. Recruiters retain responsibility for reviewing candidates and making selection decisions. The paper describes the proposed architecture, methodology, technology options, evaluation plan, limitations, and safeguards related to privacy, bias, and transparency. No measured accuracy or recruitment improvement is claimed until the system is implemented and evaluated on an appropriate dataset.

Keywords: Artificial Intelligence, Resume Screening, Natural Language Processing, Skill Extraction, Candidate Ranking, Job Matching, Recruitment Automation

I. INTRODUCTION

Background.
Recruitment is an information-intensive process. A job opening may attract resumes with different layouts, writing styles, levels of detail, and terminology. Recruiters must identify applicants whose qualifications align with the role while also considering experience, education, and job-specific requirements. When the number of applications is large, manual screening can become repetitive and time-consuming.

Problem Statement.
Traditional keyword-based filtering can miss qualified applicants who describe the same skill using different terms. For example, a resume may mention " 자연 language processing" or "NLP," while a job description uses only one expression. Simple keyword counts can also overvalue repeated terms and fail to understand whether a skill appears in a project, a course, or meaningful work experience. A structured system is needed to extract information and compare it with role requirements more consistently.

Proposed Solution.
The proposed AI-Based Resume Filter System processes resumes and a target job description to produce a ranked list of candidate profiles. It combines information extraction with text similarity and requirement-based matching. The system can identify skills, education, experience, certifications, and project information, then show which job requirements are supported by each resume. The output is a recommendation for recruiter review, not a final hiring decision.

Objectives and Scope.
The main objectives are to accept common resume formats, extract readable text, normalize candidate information, identify relevant skills and qualifications, compare candidates with a job description, and present ranked results through a dashboard. The initial project scope is resume-to-job matching for a defined set of roles. It does not attempt to infer personality, protected characteristics, or a candidate's overall worth from a document.

 This work is licensed under a Creative Commons Attribution 4.0 International License

II. RELATED WORK

Keyword-Based Resume Screening.
Early applicant tracking and resume filtering approaches often rely on keyword search, Boolean conditions, and manually configured rules. These methods are easy to understand and can work well when job requirements are explicit. However, they are sensitive to wording differences, spelling variations, abbreviations, and resume formatting. A candidate may have relevant experience without using the exact phrase selected by the recruiter.

Natural Language Processing.
Natural Language Processing provides techniques for converting unstructured text into representations that software can process. Common preprocessing operations include tokenization, normalization, stop-word handling, and phrase recognition. Named-entity recognition and rule-based patterns can help identify organizations, dates, qualifications, and other entities. Skill extraction can use a curated skills dictionary, phrase matching, or a trained model, depending on the available data and project scope.

Text Similarity and Semantic Matching.
TF-IDF represents documents using term importance and can be compared using cosine similarity. It is a useful and interpretable baseline, but it mainly captures lexical overlap. Sentence embeddings and transformer-based language models can represent semantic relationships and may identify related phrases even when they do not share the same words. These approaches require careful evaluation because semantic similarity alone does not prove that a candidate meets a mandatory qualification.

III. SYSTEM ARCHITECTURE

Architecture Overview.
The system is organized as a sequence of modules: resume upload, document text extraction, text cleaning, information and skill extraction, job-description processing, candidate matching, ranking, and dashboard presentation. Separating these modules makes the application easier to test and allows the team to replace one NLP component without redesigning the entire system.

Resume Upload and Text Extraction.
The upload component accepts supported PDF and DOCX files and validates file type and size before processing. A PDF parser or DOCX reader extracts text while preserving useful structure where possible. Scanned documents may require Optical Character Recognition, which introduces additional errors and should be treated as a separate processing path. The original document should not be modified during extraction.

Human-in-the-Loop Recruitment.
Decision-support systems can help organize information, but hiring decisions involve context that may not be present in a resume. A recruiter may need to verify employment dates, assess portfolio evidence, or consider equivalent experience. Therefore, an automated ranking should be accompanied by evidence from the resume and should allow a human reviewer to correct extraction errors, adjust job requirements, and make the final decision.

Resume Parsing and Feature Extraction.
The parser identifies sections such as skills, education, work experience, projects, and certifications. It extracts candidate attributes including skill phrases, qualification names, organization names, role titles, and date ranges when available. A normalization layer maps equivalent terms to a consistent vocabulary, for example mapping common abbreviations to their expanded forms. Each extracted item should retain its source text or page location so the reviewer can verify it.

Job Description Processing.
The job description is processed using a similar text-cleaning and extraction pipeline. Requirements are separated into mandatory skills, preferred skills, experience expectations, education requirements, and role responsibilities. The recruiter should be able to edit these extracted requirements because job descriptions may contain ambiguous wording or requirements that are not suitable for automatic interpretation.

Matching, Ranking, and Review.
The matching engine compares each candidate profile with the normalized job requirements. It may calculate skill coverage, relevant experience indicators, education alignment, and text similarity. The ranking module combines these signals using documented weights or a trained model if labelled examples are available. The dashboard displays the

overall match score together with matched requirements, missing requirements, and evidence snippets. Recruiters review this information before taking any action.

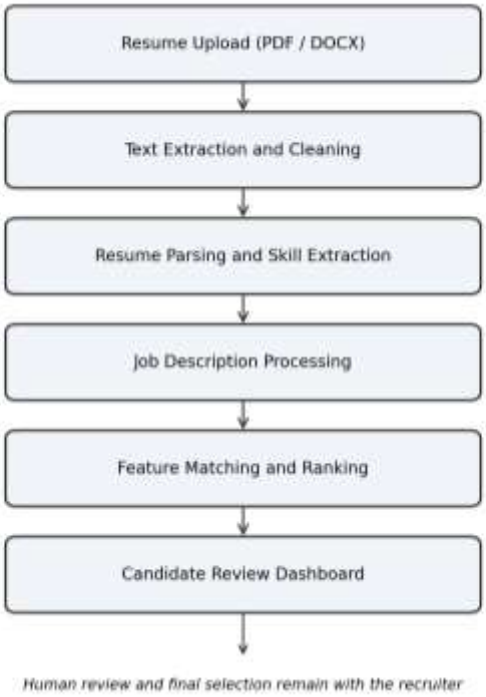


Fig. 1. Proposed architecture of the AI-based resume filtering and job-matching system.

IV. PROPOSED METHODOLOGY

Data Collection and Preparation.
The system requires a collection of sample resumes and corresponding job descriptions for development and testing. Personal information should be removed or masked where it is not needed. The team should define the target roles and the meaning of a relevant match before preparing labels. If recruiters or domain reviewers label examples, they should use a shared guideline so that the labels are applied consistently.

Document Preprocessing.
Extracted text is normalized by removing unnecessary whitespace, correcting common encoding issues, and standardizing case where appropriate. Tokenization separates text into words or phrases. Skill phrases should be preserved as multi-word expressions, since splitting a phrase such as “machine learning” into unrelated tokens can reduce matching quality. Dates and experience durations require separate parsing and validation.

Skill and Qualification Extraction.
A baseline implementation can use a curated skill taxonomy and phrase-matching rules to identify technical and domain skills. Synonyms and abbreviations can be stored in a mapping table. More advanced versions may use named-entity recognition or a transformer-based model to identify entities in context. Extracted skills should be associated with evidence text, and uncertain extractions should be flagged rather than silently treated as confirmed facts.

Candidate Feature Construction.
For each candidate, the system creates a structured profile containing normalized skills, education details, relevant experience indicators, certifications, and project evidence. For the job, it creates a structured requirement profile. Features can include the proportion of required skills matched, preferred-skill coverage, relevant experience range, and TF-IDF or embedding similarity between resume and job-description text. Features should be designed to avoid counting the same evidence multiple times.

Scoring and Ranking.
A transparent baseline score can combine weighted components such as required-skill coverage, preferred-skill coverage, relevant experience, and text similarity. Mandatory requirements may be displayed separately rather than hidden inside a single score. The weights must be configurable and documented; they should not be presented as objectively correct

without evaluation. If labelled training data becomes available, a supervised ranking or classification model can be compared with the baseline.

Output and Feedback.

The system sorts candidates by the configured matching score and displays the factors that contributed to the result. Recruiters can mark extraction errors, confirm or reject matches, and record reasons for overrides. Feedback can be used to improve the taxonomy and matching logic, but it should not automatically become training data without quality checks and governance.

V. TECHNOLOGY COMPONENTS

Programming Language and Document Processing.

Python is a suitable implementation language because it provides libraries for document processing, text analysis, machine learning, and web application development. PDF and DOCX extraction libraries can be selected according to the formats supported by the project. The extraction layer should handle unreadable files, password-protected documents, unsupported formats, and documents containing only scanned images.

Natural Language Processing Libraries.

NLTK or spaCy can support tokenization, phrase processing, and entity extraction. A curated skill dictionary can be maintained as a CSV or database table to support synonym normalization. Scikit-learn can provide TF-IDF vectorization, cosine similarity, and baseline machine-learning models. If semantic matching is required, sentence-transformer models may be evaluated as an additional component.

Database and Application Interface.

SQLite can store job descriptions, extracted candidate attributes, matching results, and review decisions for a small prototype. A web interface built with Streamlit or Flask can allow a recruiter to upload resumes, select a job description, inspect ranked candidates, and review evidence. Access controls should limit who can view candidate documents and personal information.

Suggested Module Structure.

A modular implementation can separate file validation, document extraction, text preprocessing, skill extraction, job requirement parsing, feature construction, scoring, database operations, and dashboard presentation. This structure allows each component to be tested independently. It also makes it easier to compare a keyword-based baseline with a semantic similarity approach without changing the upload or review interface.

VI. EVALUATION FRAMEWORK

Dataset and Test Design.

Evaluation should use a representative set of resumes and job descriptions covering the roles the project intends to support. The dataset should include varied resume layouts, different ways of describing similar skills, and applicants with different experience levels. If synthetic or publicly available sample resumes are used, their origin and limitations should be stated. Data used for model development should be separated from the final test set.

Matching Quality Metrics.

If the system predicts whether a resume is relevant to a job, precision, recall, and F1-score can measure classification quality. Precision indicates how many shortlisted resumes are truly relevant according to the reference labels, while recall indicates how many relevant resumes were retrieved. For a ranked list, Precision@K, Recall@K, and ranking metrics such as nDCG may be useful. The choice of metric should match the intended recruiter workflow.

Extraction Quality.

Skill extraction can be evaluated by comparing automatically extracted skills with manually annotated examples. Precision measures the correctness of extracted items, while recall measures how many relevant items were found. The team should also inspect errors caused by tables, columns, unusual fonts, abbreviations, and scanned pages. A strong ranking result can still be misleading if the underlying extracted profile is incorrect.

Fairness and Robustness Checks.

The team should test whether equivalent qualifications expressed with different wording receive similar treatment. It should also examine whether irrelevant personal details, such as names or photographs, influence the ranking. Where lawful and ethically appropriate data is available, results can be audited across relevant groups; sensitive attributes should

not be inferred from resumes. Any detected disparity requires investigation rather than an assumption that the score is neutral.

System Performance.

Operational evaluation can measure document-processing time, throughput for a batch of resumes, dashboard response time, and failure rate for unsupported or malformed files. The final report should include dataset size, role coverage, annotation method, model or scoring configuration, software versions, and metric values. No accuracy, time-saving, or fairness result should be claimed until it has been measured.

VII. ILLUSTRATIVE CANDIDATE MATCHING EXAMPLE

Example Job Requirement.

Assume a sample software-development role requires Python, SQL, data structures, and basic web development, while machine learning is listed as a preferred skill. These requirements are entered into the job-description module and reviewed by the recruiter before matching begins.

Candidate Profile A.

A sample resume lists Python, SQL, data structures, and web development projects, along with a relevant internship. The extraction component identifies the required skills and links them to project or experience evidence. The matching engine therefore reports strong coverage of the mandatory requirements and identifies the preferred machine-learning skill as absent if it is not supported by the document.

Candidate Profile B.

Another sample resume lists Python and machine-learning projects but contains limited evidence of SQL and web development. The system may identify strong alignment with the preferred skill but lower coverage of some mandatory requirements. The dashboard should show this distinction rather than relying only on a single overall score.

Interpretation.

These examples demonstrate how the system can make its ranking more informative by separating mandatory and preferred requirements. They are hypothetical examples for explaining the workflow, not actual candidate evaluations or measured model outputs. A recruiter must verify the evidence and may consider equivalent experience that the parser did not recognize.

VIII. LIMITATIONS, PRIVACY, AND ETHICAL CONSIDERATIONS

Data Quality and Resume Variation.

Resume formats vary considerably. Text extraction may fail on scanned pages, complex tables, multi-column layouts, images, or unusual fonts. A parser may also misinterpret employment dates or confuse a skill mentioned in a project description with a skill used professionally. The interface should expose extraction evidence and allow corrections.

Bias and Discrimination Risks.

Historical hiring data may reflect past preferences or structural inequalities. A model trained on such data can reproduce those patterns. The system should not use protected characteristics or proxies as ranking criteria, and developers should examine whether apparently neutral features create unjustified disadvantages. Automated scores should never be treated as proof that a person is suitable or unsuitable.

Privacy and Security.

Resumes contain personal data such as contact details, education history, employment records, and sometimes photographs or addresses. The system should collect only information necessary for the stated screening purpose, restrict access, protect stored files, and define a retention and deletion policy. Candidate documents should not be shared with external services unless the organization has an appropriate legal basis and user-facing disclosure.

Transparency and Human Oversight.

Candidates and recruiters should be able to understand the role of automated screening. The dashboard should show matched and missing job requirements, supporting resume evidence, and limitations of the score. Recruiters must retain the ability to review all applications, correct parsing errors, and make final decisions. The system should not automatically reject candidates solely because of a model score.

IX. FUTURE WORK

Improved Semantic Understanding.

Future versions can evaluate sentence embeddings or transformer-based models to improve matching when resumes and job descriptions use different wording. These models should be compared with transparent keyword and TF-IDF baselines. A more complex model should be adopted only if it improves relevant metrics and remains understandable enough for the intended use.

Advanced Resume Parsing.

The parser can be improved to recognize section boundaries, employment timelines, project descriptions, certifications, and skill proficiency evidence. Optical Character Recognition can be added for scanned documents. A confidence indicator and manual correction interface can help reduce the impact of uncertain extraction.

Explainable Ranking and Recruiter Feedback.

The ranking engine can provide a breakdown of score components and evidence snippets. Recruiter feedback can help identify missing synonyms or inaccurate requirement extraction. Any learning from feedback should use reviewed labels, documented model versions, and a process for monitoring unintended effects.

Deployment and Integration.

The project may later integrate with an applicant tracking system, role management portal, or secure cloud document store. A production deployment would require authentication, authorization, encryption, audit logging, secure file handling, monitoring, and a documented data-retention process. Integration should be introduced only after the prototype passes functional, security, and fairness testing.

X. CONCLUSION

Summary.

This paper proposed an AI-based Resume Filter System to assist recruiters in organizing and comparing candidate applications against job requirements. The system extracts resume text, identifies skills and qualifications, processes job descriptions, constructs candidate features, and ranks profiles according to documented matching criteria. Its modular architecture supports a baseline implementation using NLP techniques and can be extended with semantic models when suitable evaluation data is available.

Expected Contribution.

The project aims to reduce repetitive manual comparison and make candidate-job alignment easier to inspect. Its key contribution is not simply a ranked list, but a review interface that shows which requirements were matched, which were not found, and what resume evidence supports the result. This makes the output more useful and easier to challenge than an unexplained score.

Final Note.

Resume filtering is a high-impact application because its results can influence access to employment opportunities. Therefore, privacy, fairness, transparency, and human oversight are essential parts of the system rather than optional additions. The proposed design should be implemented and evaluated carefully before any real recruitment use. Actual effectiveness must be established through a documented dataset, appropriate metrics, and review by people familiar with the relevant roles.

ACKNOWLEDGMENT

The authors acknowledge the guidance and academic support provided for the Machine Learning project.

REFERENCES

[1]. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. NAACL-HLT, 2019.
[2]. T. Joachims, “Optimizing Search Engines Using Clickthrough Data,” in Proc. ACM SIGKDD, 2002.
[3]. C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.
[4]. F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[5]. spaCy, “Industrial-strength Natural Language Processing in Python.” [Online]. Available: https://spacy.io/
[6]. scikit-learn developers, “Feature Extraction and Text Similarity Documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/feature_extraction.html
[7]. Streamlit, “Streamlit Documentation.” [Online]. Available: https://docs.streamlit.io/
[8]. NIST, “Artificial Intelligence Risk Management Framework (AI RMF 1.0),” 2023. [Online]. Available: https://www.nist.gov/itl/ai-risk-management-framework