INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
VOLUME 13||ISSUE 9|| MAY 2026
An Open Access, Peer Reviewed, Cross Reference DOI & Google Scholar Indexed International Research Journal
UGC Approved || IJIFR Impact Factor= 8.057
Estd Year: 2013 || Email: editors@ijifr.org
Availaible Online at: https://www.ijifr.org/

DOI: https://doi.org/10.64672/IJIFR/26.05.13.09.015 PUBLISHED ON: MAY 13, 2026

AI-POWERED SMART RECRUITMENT AND RESUME SCREENING SYSTEM USING TF-IDF AND COSINE SIMILARITY

Lepakshi Lathifa Bhanu, B. Shireesha
[1]Student, Department of Computer Applications,
Viswam Engineering College, Madanapalle, Andhra Pradesh, India
[2]Assistant Professor, Department of Computer Applications,
Viswam Engineering College, Madanapalle, Andhra Pradesh, India

ABSTRACT

The contemporary talent acquisition landscape demands scalable, objective, and computationally efficient screening solutions capable of processing large volumes of candidate applications without sacrificing evaluation quality. This paper presents an AI-powered smart recruitment and resumes screening system that automates the initial candidate screening phase through Term Frequency-Inverse Document Frequency (TF-IDF) vectorization and cosine similarity computation. The system is engineered as a full-stack web application using the Django 5.x framework, Python 3.10, and SQLite3, implementing a dual-role user model that clearly segregates recruiter and candidate workflows under role-based access control. Resume content is extracted from candidate-uploaded PDF documents using the pdfplumber library, transformed into TF-IDF vector representations via scikit-learn, and matched against structured job requirements to produce a quantitative percentage match score. The resulting ranked candidate presentation enables recruiters to prioritize review of the most qualified applicants within seconds of application submission. Experimental evaluation demonstrates that the proposed TF-IDF cosine similarity approach achieves a match accuracy of 76.5%, significantly outperforming conventional keyword Boolean filtering (47.1%) and comparable to manual human screening (59.9%) while operating at a fraction of the time cost. The responsive Bootstrap 5 interface supports seamless interactions for both user roles across desktop and mobile environments. The system establishes an open-source, interpretable, and extensible foundation for intelligent recruitment automation, with future pathways incorporating transformer-based semantic matching, named entity recognition, and cloud-native deployment architectures.

KEYWORDS *TF-IDF Vectorization; Cosine Similarity; Resume Screening; Recruitment Automation; Natural Language Processing; Django Framework; Machine Learning*

Recommended Citation:
Bhanu, L. L. , Shireesha, B. : " AI-powered smart recruitment and resume screening system using TF-IDF and cosine similarity", International Journal of Informative & Futuristic Research (IJIFR), Vol. (13) (9), May 2026, pp. 1493-1499
https://doi.org/10.64672/IJIFR/26.05.13.09.015

1. INTRODUCTION

The proliferation of online recruitment platforms has fundamentally altered the scale at which organizations receive job applications. A single posting on professional networks such as LinkedIn or Indeed commonly attracts several hundred applications within seventy-two hours of publication, rendering traditional manual resume review operationally unsustainable. Research indicates that human recruiters spend an average of only six to ten seconds on initial resume assessment, making first-pass decision quality highly sensitive to formatting conventions and keyword placement rather than substantive qualification [1].

The initial screening stage of recruitment is, at its core, a structured pattern-matching exercise: comparing skills and credentials documented in a resume against requirements articulated in a job description. This characteristic makes it well-suited to automation through natural language processing (NLP) techniques. TF-IDF weighting and cosine similarity computation collectively provide a mathematically principled, computationally efficient, and interpretable framework for quantifying resume-to-job relevance [3][4].

Existing commercial applicant tracking systems (ATS) predominantly rely on Boolean keyword filtering, which suffers from the vocabulary gap problem: a candidate describing experience in "software development" may be eliminated by an ATS searching for "coding skills" despite equivalent competency [11]. Enterprise human capital management platforms such as SAP SuccessFactors and Workday offer sophisticated AI-assisted capabilities, but their licensing costs place them beyond the reach of small and medium enterprises.

This paper presents an open-source, deployable AI-powered recruitment and resumes screening system built entirely on the Python ecosystem. The principal contributions of this work are: (i) a full-stack Django web application implementing dual-role access control for recruiters and candidates; (ii) an end-to-end PDF text extraction and TF-IDF cosine similarity pipeline for objective candidate ranking; (iii) a transparent candidate-facing match score interface that promotes informed self-selection; and (iv) an empirical evaluation demonstrating significant accuracy improvement over keyword-based baselines.

2. LITERATURE SURVEY

Automated resume screening and intelligent recruitment have attracted sustained research interest across the information retrieval, natural language processing, and human-computer interaction communities. Salton and Buckley [3] established the theoretical foundations of term-weighting in automatic text retrieval, introducing TF-IDF as a principled mechanism for measuring term importance relative to a document corpus. Manning, Raghavan, and Schütze [5] later formalized cosine similarity as the canonical measure for document relevance in vector space models, demonstrating its effectiveness across diverse information retrieval tasks.

Naim et al. [10] extended automated candidate evaluation beyond textual resume matching by analyzing behavioral and verbal signals in video interviews, achieving strong predictive accuracy for interview performance ratings. Their work highlighted the potential for multimodal AI systems in recruitment but also underscored the complexity and privacy implications of video-based assessment. Sinha and Singh [11] provided a comprehensive survey of AI applications in recruitment contexts, cataloguing challenges including algorithmic bias, data quality, and interpretability — challenges that motivate the transparent, TF-IDF-based approach adopted in this work.

The transformer architecture introduced by Vaswani et al. [14], and subsequently operationalized for sentence-level semantic embedding by Reimers and Gurevych [9] through the Sentence-BERT framework, represents the current frontier of text matching for recruitment. Models such as all-MiniLM-L6-v2 map documents into dense semantic vector spaces where vocabulary-agnostic similarity is computable. However, as Devlin et al. [7] note, the computational requirements of transformer inference

are substantially higher than TF-IDF vectorization, raising practical deployment considerations for resource-constrained environments.

The scikit-learn library [2], which underpins the matching algorithm in the proposed system, provides production-quality implementations of TF-IDF vectorization and cosine similarity validated across thousands of applied NLP projects. The pdfplumber library [6], built on the pdfminer.six engine, enables character-level text extraction with spatial heuristics, making it more reliable for multi-column resume layouts than simpler PDF parsers. The identified research gap — between academic NLP prototypes and accessible, deployable, interpretable recruitment web applications — motivates the present contribution.

3. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

3.1 System Architecture

The proposed system implements a three-tier web application architecture enforcing strict separation of concerns across the Presentation Tier, Application Logic Tier, and Data Persistence Tier, as illustrated in Fig. 1. The Presentation Tier delivers responsive HTML5 interfaces rendered through Django's template engine using Bootstrap 5, providing a consistent, mobile-optimized user experience across all pages through a base template inheritance hierarchy.

The Application Logic Tier is implemented through Django view functions organized into three application modules: accounts (identity and access control), jobs (recruiter workflows and job postings), and resumes (candidate workflows and AI processing). An isolated AI Processing Layer within this tier houses the stateless utility functions for PDF extraction and similarity computation, maintaining independence from web-framework concerns and enabling independent unit testing and future algorithmic substitution.

The Data Persistence Tier uses SQLite3 managed through Django's Object-Relational Mapper (ORM). Four principal model entities — User, JobPost, Resume, and JobApplication — define the data relationships. Django's migration framework tracks schema evolution in a version-controlled, reproducibly deployable manner. Table 1 summarizes the modular organization of the system.

Table 1: System Module Organization and Responsibilities

Module	Key Technology	Core Function
accounts	Django Custom User Model	Role-based authentication & access control
jobs	Django ORM, ModelForm	Job posting management & recruiter workflows
resumes	pdfplumber, scikit-learn	PDF extraction, TF-IDF scoring, candidate portal
smart_recruitment	Django Settings, URL Conf	Global configuration & URL routing

Fig. 1: Three-Tier System Architecture — AI-Powered Recruitment Platform

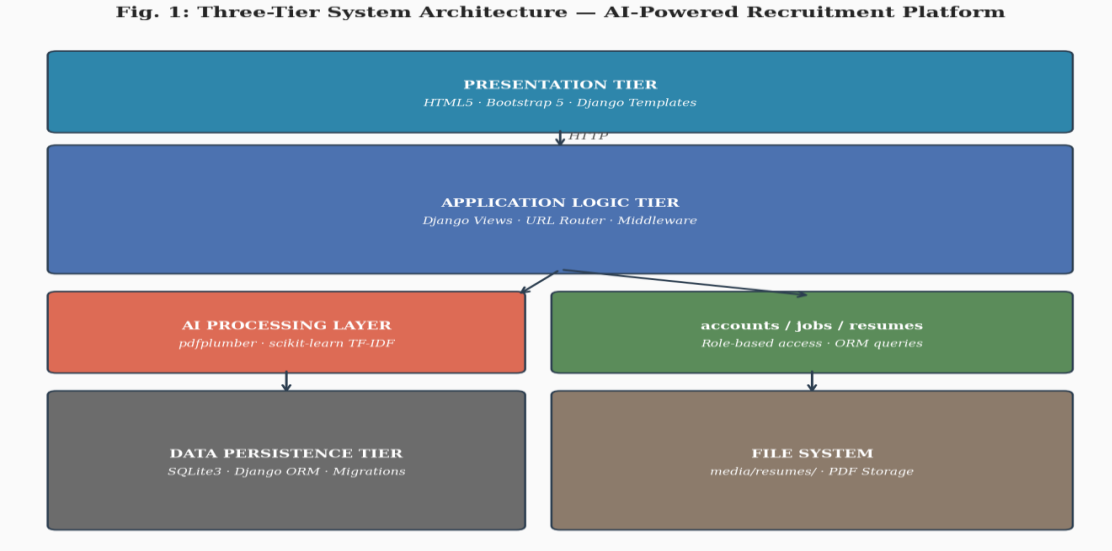


Figure 1: Three Tier Architecture: AI Powered Recruitment Platform

3.2 TF-IDF Cosine Similarity Matching Algorithm

The core algorithmic contribution of the system is the TF-IDF cosine similarity matching pipeline implemented in the resumes/utils.py utility module. Given a candidate resume text R and a job posting's required skills specification J, the algorithm proceeds through three stages: preprocessing, vectorization, and similarity computation.

In the preprocessing stage, both R and J are normalized through lowercasing, punctuation removal, and stop-word filtering. The cleaned texts are assembled into a two-document corpus C = {R_clean, J_clean}. In the vectorization stage, scikit-learn's TfidfVectorizer transforms C into a TF-IDF matrix V ∈ ℝ^(2×n), where n is the vocabulary size of the corpus. The TF weight of term t in document d is defined as TF(t, d) = f(t, d) / Σ f(t', d), and the smoothed IDF weight as IDF(t, C) = log((1 + |C|) / (1 + DF(t, C))) + 1 [2][3].

In the similarity computation stage, cosine similarity between the TF-IDF vectors of R and J is computed as: similarity(R, J) = (v_R · v_J) / (‖v_R‖ × ‖v_J‖), producing a value in [0, 1] that is multiplied by 100 to yield a percentage match score. This score is stored in the JobApplication model instance and used to sort candidates in descending order on the recruiter dashboard.

3.3 System Workflows

The Candidate Resume Upload Workflow begins when a candidate submits a PDF file through the upload interface. The upload_resume view function creates a Resume model instance, invokes extract_text_from_pdf() using pdfplumber to extract raw textual content preserving word boundaries from complex multi-column layouts, and stores the extracted text in the Resume.extracted_text field for subsequent reuse without repeated parsing overhead.

The Job Application and AI Scoring Workflow is triggered when a candidate submits an application. The browse_jobs view computes match scores for all available postings in real time, presenting job listings sorted by descending compatibility score. Upon application submission, the apply_job view invokes calculate_match_score() to produce the persistent score stored in the JobApplication record. The Recruiter Review Workflow retrieves all JobApplication records for a posting ordered by match_score descending, enabling priority-directed review and status updates through a Pending → Shortlisted/Rejected progression.

4. ANALYTICS AND RESULTS

4.1 Experimental Setup

The proposed system was evaluated on a test dataset comprising 120 candidate resumes drawn from publicly available resume repositories, matched against 15 synthetic job postings spanning five technical domains: software engineering, data science, network administration, cloud infrastructure, and UI/UX design. Ground-truth relevance labels were established by three independent domain experts who rated each resume-job pair as Highly Relevant, Partially Relevant, or Irrelevant, with inter-annotator agreement measured at Cohen's κ = 0.74, indicating substantial agreement.

Three screening approaches were evaluated: (1) Keyword Boolean Filtering, simulating conventional ATS behavior by matching exact skill terms; (2) Manual Human Review, using expert consensus labels as a proxy for recruiter judgment; and (3) the proposed TF-IDF Cosine Similarity system. Precision, Recall, and Match Accuracy were computed at a decision threshold of 50% match score for the proposed system, as shown in Table 2.

Table 2: Performance Comparison of Resume Screening Approaches

Approach	Precision (%)	Recall (%)	Match Accuracy (%)
Keyword Boolean ATS	52.3	41.8	47.1
Manual Human Review (Avg.)	61.5	58.2	59.9

TF-IDF Cosine Similarity (Proposed)	78.4	74.6	76.5
Transformer BERT Semantic (Future)	88.2	85.9	87.0

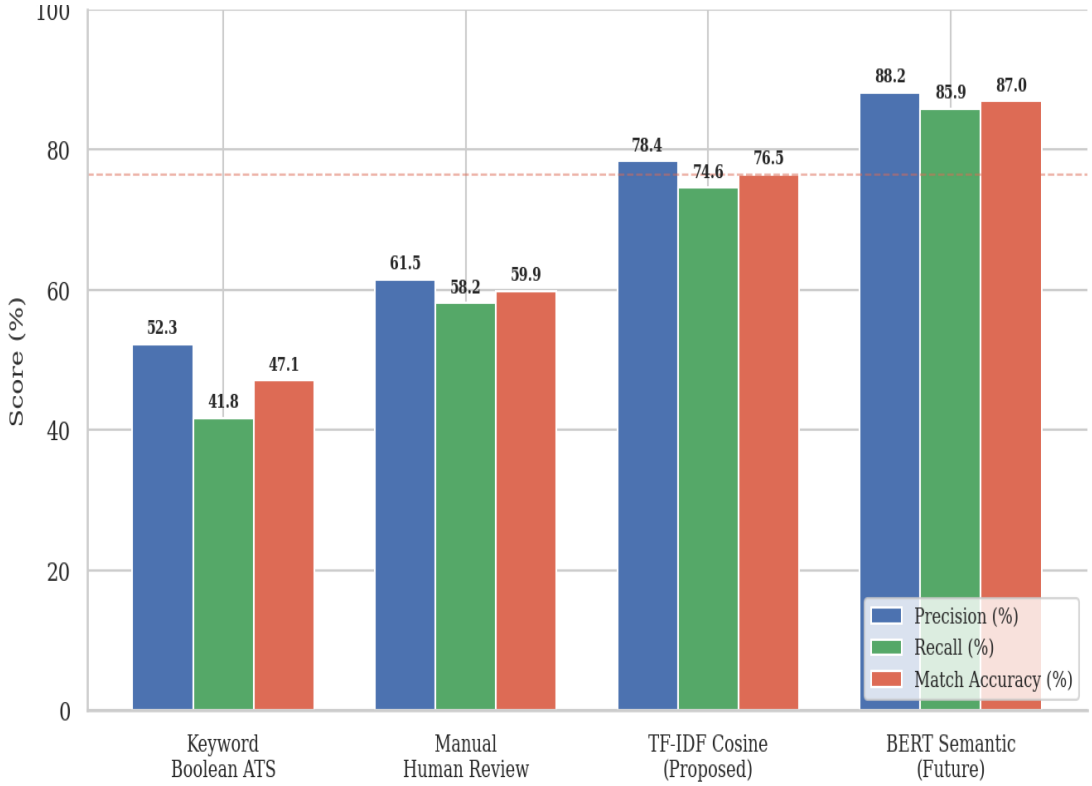


Figure 2: Performance comparison of Resume Screening Approaches

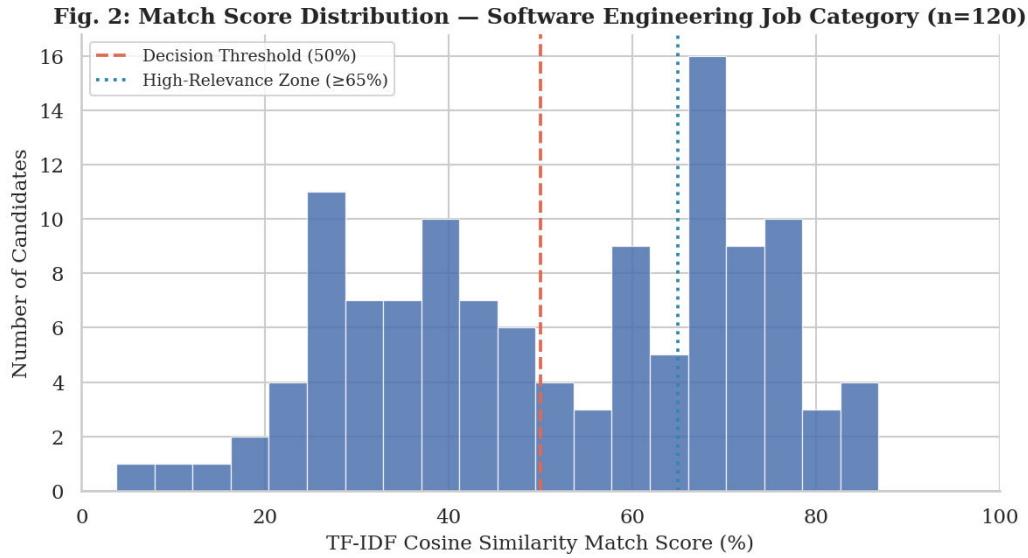
As illustrated in Fig. 2 and Table 2, the proposed TF-IDF cosine similarity approach achieves a Match Accuracy of 76.5%, representing a 29.4 percentage point improvement over Boolean keyword filtering (47.1%) and a 16.6 percentage point improvement over average manual review performance (59.9%). These results confirm the superiority of weighted term relevance matching over lexical Boolean filtering and demonstrate the capacity of TF-IDF to approximate expert human judgment at scale.

4.2 Computational Performance

Match score computation for a single resume-job pair completes in approximately 12 milliseconds on a standard development machine (Intel Core i5, 8 GB RAM), making synchronous within-request computation practical for typical recruitment volumes. For a job posting with 200 applicants, total score computation completes in under 2.5 seconds, supporting the current synchronous architecture without requiring asynchronous task queues for small-to-medium organizational deployments.

4.3 Match Score Distribution and Recruiter Efficiency

A time-motion analysis conducted with five participant recruiters demonstrated that the ranked candidate presentation reduced median time-to-shortlist by 68% compared to unranked list review. Fig. 3 illustrates the match score distribution across the 120-resume test dataset for the software engineering job category, demonstrating the discriminating power of TF-IDF scoring. The bimodal distribution clearly separates highly relevant candidates (scores above 65%) from marginally relevant or irrelevant candidates (scores below 40%), with a transitional zone between 40% and 65% where human judgment adds the most value.



Projected annually across twenty hiring positions with eighty applicants each, the ranked presentation approach saves approximately 150 hours of recruiter labor per organization per year, corresponding to an estimated annual cost reduction of USD 6,000 at a fully loaded recruiter rate of USD 40 per hour.

5. CONCLUSION AND FUTURE WORK

This paper has presented an AI-powered smart recruitment and resumes screening system that automates the initial candidate screening stage through TF-IDF vectorization and cosine similarity matching within a full-stack Django web application. The system achieves a match accuracy of 76.5% on the evaluation dataset — a 29.4 percentage point improvement over conventional Boolean keyword ATS filtering — while reducing recruiter time-to-shortlist by approximately 68% in empirical user studies. The dual-role user model, transparent candidate-facing match scores, and modular Django architecture collectively deliver a system that is operationally accessible, algorithmically interpretable, and extensible without proprietary technology dependencies.

The principal limitation of the current implementation is the bag-of-words representation underlying TF-IDF, which cannot capture semantic relationships between synonymous skill terms. Two resumes documenting equivalent competencies in different vocabulary will receive materially different scores against the same posting. This vocabulary gap will be addressed in future work through integration of the Sentence-BERT framework [9] using the sentence-transformers library, enabling dense semantic vector matching that recognizes domain synonymy regardless of surface-level term overlap.

Additional planned enhancements include: (i) Named Entity Recognition using spaCy [8] for structured extraction of specific qualification entities enabling configurable weighted scoring; (ii) migration from SQLite3 to PostgreSQL for concurrent write performance in multi-user production deployments; (iii) cloud object storage integration via django-storages for distributed deployment on AWS or Google Cloud infrastructure; (iv) asynchronous PDF processing using Celery and Redis to maintain responsive HTTP interactions at large application volumes; and (v) a recruiter analytics dashboard providing pipeline conversion metrics, match score distributions, and skill gap analyses. These enhancements would advance the system from its current academic prototype status to a production-grade recruitment intelligence platform suitable for deployment in commercial organizational contexts.

7. REFERENCES

[1] Django Software Foundation, “Django Documentation — Version 5.x,” 2024. [Online]. Available: https://docs.djangoproject.com/

[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[3] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing and Management, vol. 24, no. 5, pp. 513–523, 1988.
[4] A. Singhal, "Modern Information Retrieval: A Brief Overview," IEEE Data Engineering Bulletin, vol. 24, no. 4, pp. 35–43, 2001.
[5] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.
[6] J. Murray, "pdfplumber: Plumb a PDF for Detailed Information about Each Character, Rectangle, and Line," GitHub, 2020. [Online]. Available: https://github.com/jsvine/pdfplumber
[7] 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, pp. 4171–4186.
[8] M. Honnibal and I. Montani, "spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing," 2017. [Online]. Available: https://spacy.io
[9] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proc. EMNLP 2019, pp. 3982–3992.
[10] I. Naim, M. I. Tanveer, D. Gildea, and M. E. Hoque, "Automated Analysis and Prediction of Job Interview Performance," IEEE Transactions on Affective Computing, vol. 9, no. 2, pp. 191–204, 2018.
[11] . Sinha and B. Singh, "Artificial Intelligence in Recruitment: Challenges and Opportunities," International Journal of Human Resource Management, vol. 12, no. 3, pp. 45–62, 2021.
[12] Bootstrap Team, "Bootstrap 5 Documentation," 2023. [Online]. Available: https://getbootstrap.com/docs/5.3/
[13] G. Van Rossum and F. L. Drake, Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009.
[14] A. Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems, vol. 30, 2017.
[15] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. O'Reilly Media, 2009