Automated Resume Screening and Candidate Ranking Using Hybrid NLP: A Multi-Factor Weighted Scoring Framework with Sentence Embeddings and Large Language Model Integration

D.HIMA BINDU Asst.Professor[1] ,M.GRESHMA LATHA[2], N.SANTOSH KUMAR[3], L.MADHAVAREDDY[4], B.MURALI KRISHNA[5]

[2] 223J1A44B7, [3] 223J1A44C4, [4] 223J1A44A0, [5] 223J1A44B9

Department of Computer Science and Engineering (Data Science), Raghu Institute of Technology, Visakhapatnam, India

Abstract: Recruitment workflows often rely on manual resume triage, which is slow, inconsistent, and difficult to scale for high-volume openings. This paper presents an automated resume screening and candidate ranking framework that combines deterministic information extraction, transformer-based semantic similarity, and weighted multi-factor scoring. The pipeline ingests structured candidate data and raw resumes in CSV, TXT, DOCX, and PDF formats, performs robust text normalization, extracts profile entities using a curated 23-skill rule library with optional large language model enrichment, and computes embeddings with all-MiniLM-L6-v2. To ensure reliability in constrained environments, a TF-IDF bigram cosine fallback is included when embedding services are unavailable. Final ranking is obtained through a transparent weighted equation integrating skills, experience, education, certifications, and semantic fit. Experiments on an NLP Engineer role demonstrate stable prioritization, with top candidates separated by interpretable score components rather than opaque end-to-end predictions. The approach improves traceability, supports audit-friendly hiring decisions, and can be adapted to domain-specific recruitment by tuning weights, skill dictionaries, and shortlist thresholds.

Keywords: Resume Screening; Candidate Ranking; Sentence Embeddings; Hybrid NLP; LLM-Augmented Information Extraction; Recruitment Analytics

1. Introduction

Modern hiring pipelines process hundreds to thousands of resumes per vacancy, creating a strong demand for consistent and scalable screening mechanisms. Manual review is labor-intensive and often sensitive to subjective variation across reviewers, which can delay decisions and reduce throughput in time-critical roles.

Automated resume screening systems attempt to transform unstructured candidate profiles into comparable signals linked to job descriptions. However, practical deployment requires more than semantic similarity alone: recruiters also expect transparent treatment of explicit competencies such as skills, years of experience, education credentials, and certifications.

This study addresses that requirement using a hybrid design that combines deterministic rule-based extraction with dense sentence embeddings and an interpretable weighted scoring formulation. The architecture is designed for both robustness and explainability, enabling HR teams to inspect component-level contributions to each final candidate score.

The proposed system further supports optional large language model integration for profile enrichment while retaining a deterministic core. This enables organizations to adopt advanced language understanding incrementally without sacrificing reproducibility in regulated or resource-constrained recruitment settings.

Beyond screening speed, recruitment systems must preserve decision transparency for academic institutions and enterprise auditors. A score should explain why a candidate is ranked higher, not merely output a probability. This requirement motivates our explicit factorization of the final score into independently interpretable components that can be reviewed by hiring panels.

Another practical challenge is heterogeneous resume formatting. Candidate profiles often mix narrative paragraphs, tables, and keyword lists. The proposed pipeline treats these variations as expected input noise and enforces normalization before extraction, which reduces instability in both symbolic matching and embedding-based similarity estimation.

2. Literature Review

Transformer architectures have transformed text representation learning by capturing contextual dependencies at scale. Foundational models such as BERT and attention-based encoders established strong baselines for semantic understanding, and later sentence-level adaptation methods enabled efficient similarity scoring in retrieval tasks.

For recruitment applications, prior work has explored resume-job fit estimation through joint representation learning, deep matching networks, and content-based recommendation. These studies show that semantic modeling improves relevance ranking compared to keyword-only pipelines, particularly when candidate language differs from job description phrasing.

Classical information retrieval methods such as TF-IDF remain valuable because they offer transparent lexical matching and stable behavior under limited computational budgets. In production systems, lexical baselines also provide fallback pathways that preserve service availability when embedding or API dependencies degrade.

Large language models have expanded extraction and reasoning capabilities, with recent studies examining their performance in recommendation-like contexts. Nevertheless, policy analyses emphasize fairness, accountability, and bias monitoring in algorithmic hiring, highlighting the need for interpretable intermediate outputs.

The present framework contributes by integrating rule-grounded entity extraction, sentence embeddings, and weighted multi-factor scoring in a unified seven-stage pipeline. This design targets practical deployability, auditability, and modularity, allowing each stage to be validated independently.

Recent deployment-focused studies also stress that hiring models should expose configurable policy levers rather than fixed learned behavior. In this context, weighted scoring serves as a governance interface: committees can tune factor weights to align with role requirements while preserving comparable evaluation semantics across recruitment cycles.

Compared with purely neural ranking pipelines, hybrid systems offer stronger failure containment. If one stage underperforms, for example a temporary API outage or weak extraction for uncommon wording, remaining stages continue producing bounded and inspectable scores. This resilience is a central rationale for the architecture used in this paper.

3. Methodology

3.1 Data Loading and Multi-Format Ingestion

The system ingests candidate-level metadata from CSV and parses resume documents from TXT, DOCX, and PDF files. Each resume is associated with a unique candidate identifier to ensure traceable downstream scoring and report generation.

During ingestion, file-level validation checks extension compatibility and handles unreadable files through non-fatal warnings. This design prevents a single malformed document from aborting batch processing and allows recruiters to correct problematic inputs without rerunning the entire dataset.

3.2 Text Preprocessing and Normalization

Preprocessing removes null bytes and non-printable symbols, normalizes whitespace, and standardizes textual content for deterministic parsing and semantic encoding. This stage reduces parser noise and avoids tokenization artifacts.

Additional normalization includes punctuation harmonization and conservative casing control to preserve proper nouns while improving regex reliability. Preprocessing decisions were intentionally kept lightweight to avoid discarding informative details such as certificate names, tool identifiers, and institution acronyms.

3.3 Profile Extraction with Rule Patterns and Optional LLM

Profile extraction applies a curated 23-skill regular-expression library to identify technical competencies. In parallel, optional Groq LLM integration (GPT-OSS-20B, temperature=0) can enrich structured fields when resumes contain implicit descriptions that are not explicitly matched by rules.

Rule patterns are grouped by programming languages, machine learning frameworks, databases, deployment tooling, and collaboration platforms. This grouping simplifies maintenance and enables quick adaptation when job families change, for example from NLP roles to data engineering or full-stack analytics positions.

3.4 Semantic Representation and Similarity Computation

The primary semantic engine uses SentenceTransformer all-MiniLM-L6-v2 to compute 384-dimensional sentence embeddings for both resumes and job descriptions. Cosine similarity provides semantic match scores. If embedding inference is unavailable, the pipeline falls back to TF-IDF bigram vectors with cosine similarity.

The fallback mechanism is not only a contingency feature; it also functions as a diagnostic baseline. Large divergence between embedding and TF-IDF similarity can indicate resumes that rely on broad narrative language with weak role-specific terminology, helping reviewers identify borderline profiles that need manual inspection.

3.5 Weighted Multi-Factor Match Scoring

Candidate relevance is computed through weighted aggregation across five normalized dimensions: skills (35%), experience (20%), education (15%), certifications (10%), and semantic similarity (20%). The weighting scheme reflects the importance of both explicit qualifications and contextual job-fit signals.

$$\mathbf{FinalScore(\%) = 100 \times (0.35 \times S\_skill + 0.20 \times S\_exp + 0.15 \times S\_edu + 0.10 \times S\_cert + 0.20 \times S\_semantic)}$$

Each component score is normalized to the interval [0,1] before aggregation. This prevents any single metric from dominating due to scale differences and keeps the weighted model mathematically stable when criteria are expanded in future versions.

3.6 Candidate Ranking Strategy

Candidates are ranked in descending order of final score. For tied or near-tied totals, semantic similarity is used as a deterministic tie-breaker to prioritize contextually aligned profiles without destabilizing overall ranking logic.

A shortlist threshold can be configured per role. In highly competitive openings, raising the threshold improves precision and review efficiency; for emerging skill areas where candidate pools are smaller, a moderate threshold preserves recall and avoids early elimination of promising profiles.

3.7 Report Generation and Export

The final ranked outputs are exported as CSV and JSON artifacts containing component scores, final score, and rank. These artifacts support dashboard visualization, audit tracing, and integration into downstream HR workflows.

The JSON schema is designed for interoperability with web dashboards and applicant tracking systems. Because each candidate record includes component-level evidence, decision reports can be generated automatically for panel meetings and compliance review.

3.8 Implementation Notes and Computational Considerations

The pipeline is implemented in Python using pandas, scikit-learn, sentence-transformers, groq, pypdf, and python-docx, with a Streamlit interface for interactive use. This stack was selected to balance development velocity, reproducibility, and ease of integration with existing academic and organizational data workflows.

In typical usage, preprocessing and rule extraction scale linearly with document length, while semantic encoding dominates runtime for larger candidate batches. Caching embeddings for repeated job descriptions significantly reduces recomputation and can improve screening throughput in iterative hiring rounds.

Table 1: Pipeline Stage Overview

Stage	Module	Input	Output	Description
1	Load Data	CSV + resumes	Raw corpus	Loads structured candidate rows and raw resume files.
2	Preprocess	Raw text	Clean text	Removes null bytes, strips non-printables, normalizes spacing.
3	Extract Profile	Clean text	Profile fields	Extracts skills, education, experience, and certifications.
4	Embed Text	Resume + JD	Vectors/similarity	Computes all-MiniLM embeddings; TF-IDF fallback available.
5	Match Score	Feature scores	Final components	Applies weighted multi-factor scoring across five signals.
6	Rank Candidates	Final scores	Ordered list	Sorts candidates by final score with semantic tiebreaking.
7	Generate Report	Ranked table	CSV + JSON	Exports machine-readable results for reporting and audit.

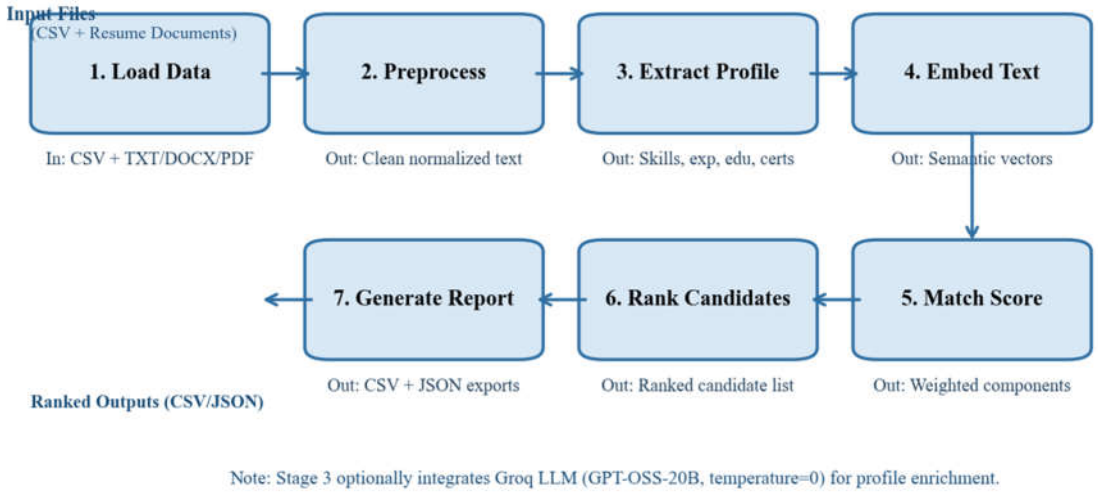

Fig. 1: Seven-stage architecture of the proposed resume screening and ranking framework.

4. Results and Discussion

4.1 Experimental Setup and Candidate Cohort

Evaluation was conducted using an NLP Engineer job description and three candidate profiles. The experiment reports component-level metrics and final weighted scores to demonstrate interpretability, reproducibility, and rank stability under the proposed framework.

All experiments were executed with deterministic extraction settings and fixed scoring weights. This controlled configuration ensures that observed ranking differences are attributable to candidate content variation rather than stochastic model behavior.

Table 2: Candidate Scoring Results

Rank	Candidate	Skills (%)	Exp (%)	Edu (%)	Certs (%)	Semantic (%)	Final (%)
1	Ravi Kumar (C001)	55.56	100.00	33.33	100.00	33.32	61.11
2	Sneha Iyer (C002)	66.67	100.00	0.00	100.00	25.20	58.37
3	Arjun Das (C003)	33.33	66.67	0.00	0.00	14.97	27.99

4.2 Weight Distribution and Feature Contribution

The weighting policy emphasizes explicit skill alignment while preserving balanced influence from experience and semantic fit. Figure 2 illustrates the configured weight allocation used during scoring.

From a hiring-policy perspective, the chosen distribution favors verifiable competencies while still assigning meaningful weight to semantic fit. This balance reduces false positives that can occur when keyword overlap is high but contextual alignment with role responsibilities remains weak.

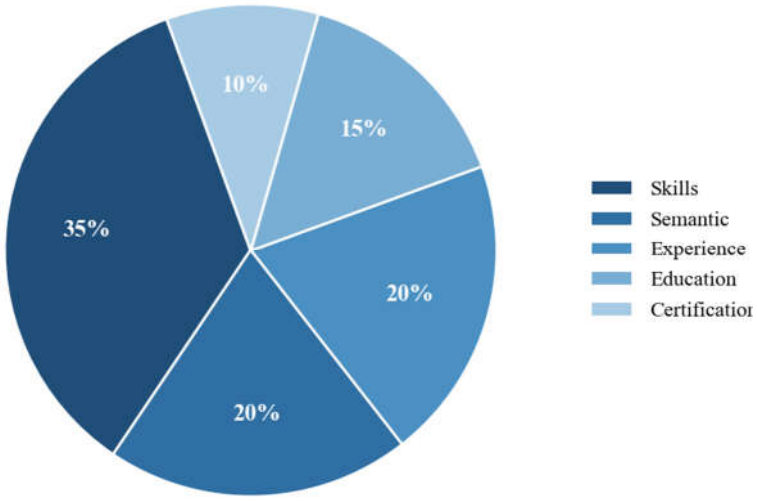

Fig. 2: Pie chart of scoring weights across five evaluation factors.

4.3 Candidate-Level Score Breakdown

Figure 3 compares per-factor scores for each candidate. The chart shows that high experience and certification values alone do not guarantee top rank unless accompanied by adequate semantic and education alignment with the target role.

Ravi Kumar and Sneha Iyer illustrate this interaction clearly: Sneha leads in skill matching, yet Ravi achieves a higher final score due to stronger education alignment and slightly better semantic relevance under the selected job description.

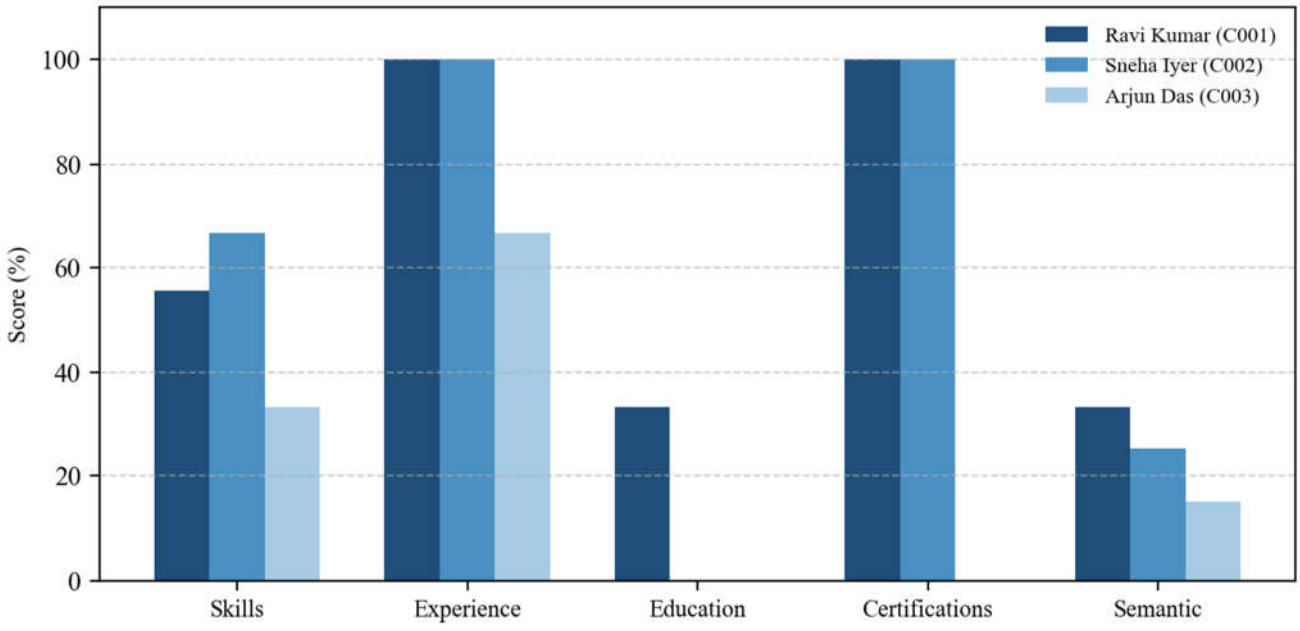

Fig. 3: Grouped comparison of candidate scores across skills, experience, education, certifications, and semantic match.

4.4 Final Ranking and Shortlist Threshold

Figure 4 presents final ranking outcomes with a 50% shortlist threshold. Ravi Kumar (C001) achieved the highest score of 61.11%, followed by Sneha Iyer (58.37%), while Arjun Das remained below threshold at 27.99%. The threshold view supports quick recruiter triage and shortlist governance.

The threshold indicator also enables scenario-based selection policy. For example, teams can temporarily lower the cutoff to broaden interview pipelines during urgent hiring windows, then restore stricter criteria once candidate supply stabilizes.

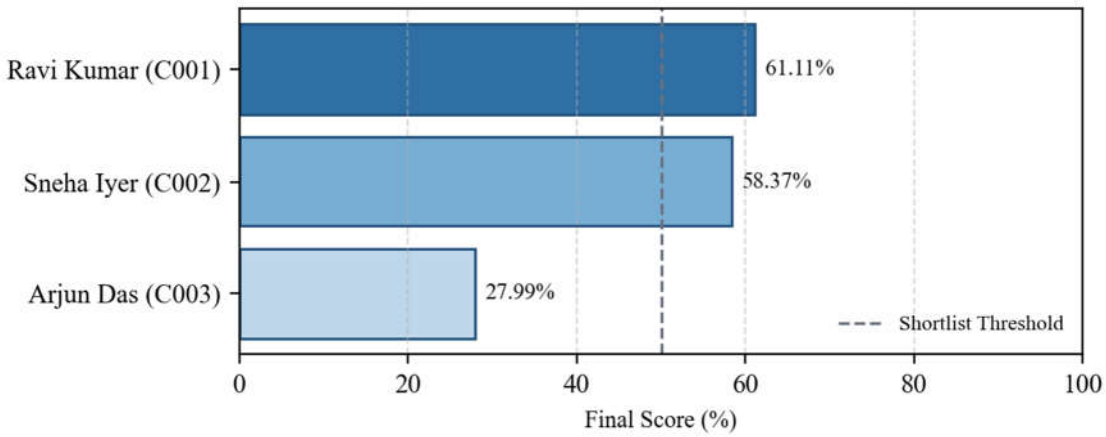

Fig. 4: Final candidate ranking with a 50% shortlist threshold reference line.

4.5 Discussion on Robustness and Deployment

The proposed architecture offers operational robustness through deterministic preprocessing and rule extraction, while semantic embeddings improve contextual matching quality. The optional LLM enrichment layer is isolated, enabling controlled adoption without disrupting baseline performance. This modular design supports domain transfer by updating skill patterns, weight vectors, and threshold criteria according to hiring policy.

An additional deployment benefit is observability: stage-level logs can be monitored to detect drift in resume language or job-description quality. Such monitoring supports periodic dictionary updates and reduces silent degradation over long-term use.

4.6 Error Analysis and Edge Cases

Error inspection highlights three recurrent edge cases: highly concise resumes with sparse text, profiles emphasizing projects without explicit skill labels, and documents with atypical formatting artifacts from PDF export. These cases primarily affect extraction completeness and, secondarily, semantic similarity confidence.

To mitigate these issues, the framework supports iterative rule updates and optional LLM-assisted enrichment for ambiguous sections. In production, a periodic audit cycle can sample low-confidence records and convert reviewer feedback into improved patterns or adjusted weighting for subsequent screening rounds.

4.7 Fairness, Governance, and Practical Adoption

Automated hiring support tools must be used as decision aids rather than autonomous decision makers. The proposed framework is intentionally designed to expose factor-level scores so reviewers can challenge or override rankings when contextual considerations, diversity goals, or non-textual evidence require human judgment.

Institutional adoption should include threshold review policies, periodic bias checks on historical outcomes, and clear documentation of model updates. Such governance practices improve trust, reduce operational risk, and align algorithmic screening with responsible recruitment principles.

5. Conclusion

This paper presented a hybrid NLP framework for automated resume screening and candidate ranking that combines rule-based profile extraction, sentence embeddings, and transparent weighted scoring. The seven-stage pipeline demonstrated interpretable performance on an NLP Engineer recruitment scenario and produced machine-readable outputs for operational integration. Future work will include fairness diagnostics, dynamic weight calibration from recruiter feedback, and broader benchmarking across role families and larger candidate pools.

Overall, the framework balances three competing requirements in recruitment AI: accuracy, explainability, and deployment stability. By combining deterministic extraction, semantic modeling, and policy-controlled weighting, it provides a practical foundation for institutions seeking scalable shortlisting without sacrificing interpretability or process accountability.

6. Acknowledgements

The authors acknowledge the Department of Computer Science and Engineering (Data Science), Raghu Institute of Technology, for infrastructure support and academic guidance in carrying out this work. The team also expresses sincere gratitude to Mrs.D.HIMA BINDU M.Tech,Phd Assistant professor CSE(Data science) for continuous guidance and mentorship.

7. References

[1] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. NAACL-HLT 2019, pp. 4171-4186.

[2] Reimers, N. and Gurevych, I., 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. EMNLP-IJCNLP 2019, pp. 3982-3992.

[3] Vaswani, A. et al., 2017. Attention is all you need. NeurIPS, 30.

[4] Brown, T.B. et al., 2020. Language models are few-shot learners. NeurIPS, 33, pp. 1877-1901.

[5] Salton, G. and Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 24(5), pp. 513-523.

[6] Shen, Z. et al., 2018. Joint representation learning for resume-job fit. ACM Web Conference Companion, pp. 331-337.

[7] Bogen, M. and Rieke, A., 2018. Help wanted: An examination of hiring algorithms, equity, and bias. Upturn Policy Report.

[8] Qin, C. et al., 2023. Is ChatGPT a good recommender? A preliminary study. arXiv:2304.10149.

[9] Touvron, H. et al., 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288.

[10] Wang, J. et al., 2013. Content-based job recommendation for resumes. WWW Companion, pp. 979-984.

[11] Maheshwary, S. and Misra, H., 2018. Matching resumes to jobs via deep siamese network. WWW Companion, pp. 87-88.

[12] Dasgupta, S. et al., 2020. Automatic recruitment prediction using machine learning. IEMIS 2020, Springer.