# TalentIQ: An Intelligent Resume & Fair Candidate Ranking System

**Dhanush C[1], Adhithyan S[2], Kamala SharathiS[3], DeepadharsanS[4], Mrs. S. Sugantha[5]**
[1, 2, 3, 4] Dept of Computer Science and Engineering
[5]Assist Prof, Dept of Computer Science and Engineering
[1, 2, 3, 4, 5] M.I.E.T Engineering College, Tiruchirappalli, Tamil Nadu, India

***Abstract-*** *TalentIQ is an AI-powered resume intelligence system designed to automate candidate screening and improve recruitment efficiency. The system analyzes resumes using Natural Language Processing (NLP) and semantic skill matching techniques to identify the most relevant candidates. Unlike traditional Applicant Tracking Systems (ATS) that rely solely on keyword matching, TalentIQ incorporates fraud detection mechanisms to identify exaggerated or misleading resume claims, including timeline inconsistencies and unsupported skill claims. The framework applies anonymization techniques to remove personally identifiable information, thereby reducing bias and ensuring fair candidate evaluation during the screening process. An explainable ranking mechanism provides transparent justifications for candidate scores, enabling data-driven hiring decisions. The system accepts resumes in PDF and DOCX formats, performs multi-factor evaluation combining skill relevance, experience validation, authenticity verification, and fraud risk indicators, and generates ranked candidate leaderboards with detailed reasoning for each evaluation decision.*

***Keywords:*** Resume Screening, Natural Language Processing, Semantic Skill Matching, Fraud Detection, Bias Mitigation, Explainable AI, Candidate Ranking, Recruitment Automation.

## I. INTRODUCTION

Modern organizations receive a large volume of resumes for each job opening, making manual screening slow, inefficient, and prone to human error. Traditional Applicant Tracking Systems (ATS) primarily rely on keyword matching and often fail to understand the semantic meaning of skills and experience. A candidate describing "Python programming" may be overlooked by a system searching for "Python development" despite identical competencies. This limitation leads to qualified candidates being filtered out while less suitable candidates progress.

Recruitment systems are also vulnerable to bias due to visible personal information such as name, gender, age, and ethnicity. Studies have demonstrated that identical resumes with different names receive significantly different callback rates, perpetuating systemic discrimination in hiring processes. Furthermore, many resumes contain exaggerated or inconsistent claims that cannot be easily detected by existing screening tools. A candidate may claim five years of experience in a technology that was only released two years ago, or list certifications that do not exist.

TalentIQ addresses these challenges through an integrated approach combining semantic understanding, fraud detection, bias mitigation, and explainable ranking. The system leverages embedding-based techniques to measure semantic similarity between candidate skills and job requirements, moving beyond surface-level keyword matching. A fraud detection module analyzes temporal relationships between education, employment history, and claimed skill maturity to identify logical inconsistencies. Anonymization removes personal identifiers before evaluation, ensuring candidates are assessed solely on qualifications. Finally, an explainable ranking mechanism provides recruiters with transparent justifications for each candidate's score, highlighting strengths, weaknesses, and detected inconsistencies.

This paper presents the complete architecture, module descriptions, implementation details, and experimental results of the TalentIQ system.

## II. RELATED WORK

### *A.Traditional Resume Screening Systems*

Traditional resume screening systems have relied heavily on rule-based keyword matching and boolean search queries. Kumar et al. [1] proposed an AI-based system that automatically analyzes resumes by extracting skills, qualifications, and experience to match candidates with job requirements. While effective for basic filtering, the approach primarily relies on keyword matching, which limits semantic understanding and cannot detect fraudulent resume claims. Similarly, Petrenko [2] presented an AI-based system for automating resume screening in the technology sector,

focusing on classifying and filtering candidates based on technical job roles. However, the system does not address bias mitigation, resume authenticity verification, or explainable decision-making.

**B.AI-Driven Resume Recommendation**

Akhtar et al. [3] developed an AI-driven intelligent resume recommendation engine that analyzes candidate resumes and recommends suitable applicants for specific job positions using NLP and machine learning. The system demonstrated improved matching accuracy compared to keyword-based approaches but exhibited limited explainability in its recommendations. The reliance on surface-level features reduces accuracy in understanding complex skills and experience, particularly when candidates describe similar competencies using different terminology.

**C. Automated Interview Systems**

Golla et al. [4] proposed an automated recruitment framework that classifies resumes using machine learning and conducts virtual AI-based interviews without human intervention. While innovative in its end-to-end automation approach, the system does not address resume authenticity verification, bias mitigation, or detailed explainable ranking mechanisms. The interview component, while valuable, cannot detect inconsistencies in the resume itself.

**D.Smart Resume Matching**

Dogra et al. [5] presented a smart resume screening system that automatically analyzes resumes and matches candidate skills with job requirements using NLP and machine learning. The system provides efficient candidate filtering but shares the limitations of keyword-based matching approaches. The absence of fraud detection and explainable ranking mechanisms limits its utility for high-stakes hiring decisions where transparency and authenticity are critical.

**E.Research Gaps**

Based on the literature survey, several research gaps remain unaddressed:

1. **Semantic Understanding:** Existing systems lack deep semantic understanding of skills and experience, treating resumes as collections of keywords rather than coherent narratives of professional capability.
2. **Fraud Detection:** No existing system incorporates automated verification of resume claims through timeline analysis and consistency checking.
3. **Bias Mitigation:** Most systems process resumes with personal identifiers visible, perpetuating demographic biases in candidate evaluation.
4. **Explainability:** Ranking decisions are typically presented without justification, reducing recruiter trust and hindering data-driven decision-making.

TalentIQ addresses each of these gaps through an integrated, multi-module architecture designed specifically for fair and transparent resume intelligence.

## III. PROBLEMSTATEMENT

Despite significant advances in natural language processing and machine learning, resume screening in real-world recruitment environments remains challenged by several practical limitations:

**Semantic Gap:** Keyword-based systems cannot distinguish between "experienced in Java programming" and "proficient in Java development," treating semantically equivalent statements as different. This results in false negatives where qualified candidates are rejected due to vocabulary mismatches.

**Fraud Vulnerability:** Resumes may contain exaggerated claims regarding employment duration, skill proficiency, educational credentials, and certifications. Manual verification is time-prohibitive, and existing automated systems lack fraud detection capabilities.

**Bias Persistence:** Personal information including name, gender, age, and ethnicity influences human evaluators and conventional systems that do not anonymize resumes before processing. This perpetuates systemic discrimination.

**Opacity of Decisions:** When a candidate is ranked poorly, recruiters cannot determine whether the low ranking resulted from skill mismatches, experience gaps, or system errors without transparent explanations.

**Scalability Constraints:** Organizations receiving thousands of applications for single positions cannot manually screen all resumes, forcing them to rely on inadequate automated filters that produce high false negative rates.

TalentIQ addresses each limitation through a unified, end-to-end pipeline that accepts raw resumes and job descriptions and returns ranked, evaluated, and explained candidate assessments.

## IV. OBJECTIVES

The specific objectives of this project are:

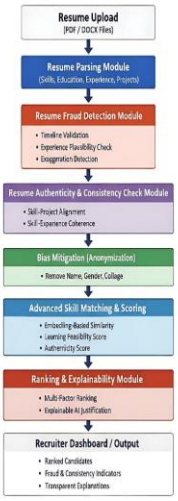
1. To design and implement an automated resume parsing system that extracts structured information—including skills, education, work experience, and certifications—from unstructured resumes in PDF and DOCX formats.
2. To develop semantic skill matching capabilities using embedding-based techniques that measure similarity between candidate qualifications and job requirements beyond keyword matching.
3. To create a fraud detection mechanism that analyzes temporal relationships between education timelines, employment history, and claimed skill maturity to identify exaggerated or inconsistent claims.
4. Toimplement bias mitigation through resume anonymization, removing personally identifiable information before evaluation to ensure fair candidate assessment.
5. To build a multi-factor scoring model that generates explainable rankings by combining skill relevance scores, experience validation results, authenticity verification outcomes, and fraud risk indicators.
6. To provide transparent justifications for each candidate's ranking, highlighting strengths, detected inconsistencies, missing skills, and overall suitability for the target role.
7. To reduce manual screening time and improve recruitment efficiency while maintaining fairness and transparency in candidate evaluation.

## V. SYSTEM ARCHITECTURE

The TalentIQ system is structured as a five-layer pipeline: a web-based frontend interface, a Flask API layer, a document processing middleware block, an AI evaluation engine, and a ranking and output generation tier. Figure 1 illustrates the complete architecture.

### A.Frontend Layer (Web Interface)

A Flask-rendered HTML/CSS/JavaScript interface allows recruiters to enter job descriptions, upload multiple candidate resumes (PDF/DOCX), and initiate the analysis process. The interface provides a job category selector, job description text area, drag-and-drop file upload zone, and a results dashboard displaying ranked candidates with detailed evaluations. A progress indicator shows processing status during analysis.




===

Fig. 1.TalentIQ System Architecture - End-to-end resume intelligence pipeline showing data flow from input to ranked output

### B.API and Routing Layer

The Flask backend exposes primary endpoints: /upload for resume file submission, /job_description for job requirement input, /analyze for initiating the evaluation pipeline, and /results for retrieving ranked candidate outputs. Request validation checks file types, sizes, and job description completeness before processing begins.

### C.*Resume Processing Middleware*

Submitted resumes are routed through a document parser that handles PDF and DOCX formats. PDF parsing uses PyPDF2 for text extraction with layout preservation. DOCX parsing utilizes python-docx to extract paragraphs, tables, and formatting. The extracted unstructured text is passed to the parsing module for structured information extraction.

### D.AI Evaluation Engine

The evaluation engine comprises five specialized modules operating in sequence:

1. **Resume Parsing Module:** Extracts structured information using spaCy NLP pipeline and custom pattern matching for skills, education, experience, and certifications.
2. **Fraud Detection Module:** Analyzes temporal relationships using date validation algorithms to detect impossible timelines (e.g., employment starting before birth, overlapping full-time positions).

3. **Authenticity Module:** Verifies skill-claim consistency by checking whether claimed skills are supported by project descriptions, work experience, or certifications.
4. **Anonymization Module:** Removes personal identifiers (name, email, phone, address, profile links) using regular expression pattern matching and named entity recognition.
5. **Skill Matching & Scoring Module:** Computes semantic similarity between candidate skills and job requirements using Sentence-BERT embeddings and cosine similarity.

### E. Ranking and Output Layer

The final layer aggregates scores from all evaluation modules, computes weighted rankings, and generates explainable outputs including candidate leaderboards, detailed critiques, missing skills identification, and performance statistics panels.

## VI. MODULE DESCRIPTION

### A. Module 1: Resume Parsing Module

**Purpose:** Extract structured information from unstructured resume documents.

**Input:** Raw resume text extracted from PDF or DOCX files.

**Process:** The module employs the spaCy natural language processing pipeline for tokenization, part-of-speech tagging, and named entity recognition. Custom pattern matching rules identify:

- **Skills:** Predefined taxonomy of technical and soft skills with fuzzy matching for variations
- **Education:** Degree names, institutions, graduation dates, and GPAs using regular expression patterns
- **Experience:** Job titles, company names, employment durations, and responsibility descriptions
- **Certifications:** Certification names, issuing bodies, and validation dates

A section detection algorithm identifies resume sections (Education, Experience, Skills, Certifications) using heading detection and content analysis.

**Output:** Structured JSON object containing categorized information with confidence scores for each extracted entity.

### B. Module 2: Resume Fraud Detection Module

**Purpose:** Detect exaggerated or inconsistent claims through temporal and logical analysis.

**Input:** Structured resume data from the parsing module, particularly dates and temporal relationships.

**Process:** The module applies three detection strategies:

1. **Timeline Consistency:** Validates that employment and education dates are chronologically possible. Flags impossible sequences such as graduation after employment start or overlapping full-time positions.
2. **Skill Maturity Analysis:** Compares claimed skill experience duration with technology age. A candidate claiming eight years of Python experience is flagged if Python's public release occurred less than eight years ago.
3. **Progression Logic:** Verifies that career progression follows plausible patterns (e.g., internship before senior position, degree before related employment).

**Output:** Fraud risk score (0-100) and detailed flags identifying specific inconsistencies with explanations.

### C. Module 3: Resume Authenticity and Consistency Check Module

**Purpose:** Verify internal consistency between claims made in different resume sections.

**Input:** Structured resume data including skills, projects, experience, and certifications.

**Process:** The module cross-references claims across sections:

- For each claimed skill, the module searches for supporting evidence in project descriptions, work experience bullet points, and certifications
- Skills lacking any supporting mention receive lower authenticity scores
- Certifications are verified against known issuing body requirements
- Education claims are checked for consistency with experience timelines

Output: Authenticity score (0-100) and evidence map showing which claims are supported and which require verification.

### D. Module 4: Bias Mitigation (Anonymization) Module

**Purpose:** Remove personally identifiable information to reduce demographic bias.

**Input:** Raw resume text and parsed structured data.

**Process:** The module applies multiple anonymization techniques:

- Regular expression patterns detect and redact email addresses, phone numbers, and physical addresses
- Named entity recognition identifies and removes person names using spaCy's NER model
- Profile links (LinkedIn, GitHub, personal websites) are detected and removed
- Potentially biased indicators (age indicators, graduation years when age-discriminatory, gender-indicating pronouns) are optionally redacted based on configuration

**Output:** Anonymized resume text preserving all qualification-relevant information while removing personal identifiers.

### *E. Module 5: Advanced Skill Matching, Scoring, and Ranking Module*

**Purpose:** Evaluate candidate suitability through semantic skill matching and generate ranked outputs.

**Input:** Job description text and anonymized, parsed candidate data.

**Process:** The module computes scores across multiple dimensions:

- **Skill Relevance Score:** Sentence-BERT embeddings generate vector representations of job requirements and candidate skills. Cosine similarity between embedding vectors produces semantic similarity scores, capturing synonymy and related terminology.
- **Experience Quality Score:** Evaluates depth and recency of experience using weighted algorithms that favor recent, relevant, and progressively responsible positions.
- **Authenticity Score:** Derived from Module 3, reflecting internal consistency of resume claims.
- **Fraud Risk Score:** Derived from Module 2, inversely weighted such that higher fraud indicators reduce overall scores.

The overall score is calculated as:

Overall Score = (0.40 × Skill Relevance) + (0.25 × Experience Quality) + (0.20 × Authenticity) + (0.15 × (100 - Fraud Risk))

**Output:** Ranked candidate list with individual dimension scores and overall rankings.

### F. Module 6: Ranking and Explainability Module

**Purpose:** Provide transparent justifications for candidate rankings.

**Input:** All scores and intermediate evaluation data from previous modules.

**Process:** The module generates human-readable explanations:

- Strengths are identified from high-scoring dimensions and well-supported skills
- Weaknesses are identified from low-scoring dimensions and missing required skills

**Output**: Candidate leaderboard with ranked positions, detailed critique for each candidate, missing skills identification, and learning feasibility scores.

## VII. ALGORITHMS

### A. Resume Text Extraction and Parsing

Resume parsing begins with document format detection. For PDF files, PyPDF2 extracts text while attempting to preserve reading order. For DOCX files, python-docx extracts paragraphs and tables sequentially. The extracted text undergoes cleaning to remove artifacts, excessive whitespace, and non-standard characters.

The cleaned text is processed using spaCy's English language model. Custom entity ruler patterns define skill, education, and certification entities. Section boundaries are detected using heading heuristics (uppercase lines, common section headers, font size variations where detectable).

### B. Semantic Skill Matching with Sentence-BERT

Traditional keyword matching fails when candidates describe skills using different terminology. TalentIQ employs Sentence-BERT (SBERT), a siamese transformer network, to generate semantically meaningful embeddings:

For a job requirement r and candidate skill s, the similarity is computed as:

Similarity(r, s) = cosine_similarity(embed(r), embed(s))

Where embed() produces a 384-dimensional normalized vector. The overall skill relevance score aggregates similarities across all job requirements, weighted by requirement importance extracted from job description frequency and phrasing.

### C. Fraud Detection through Temporal Analysis

Temporal fraud detection applies constraint satisfaction to date sequences. For each candidate, the module constructs a timeline of events (education start/end, employment start/end, certification dates). Constraints are evaluated:

∀ events i, j: if event i must precede event j, then date(i) ≤ date(j)

Violations are flagged with severity proportional to temporal inconsistency magnitude. Overlap detection for full-time employment flags positions with overlapping durations unless one is explicitly marked as part-time or freelance.

Skill maturity fraud detection compares claimed experience duration against technology age:

Claimed Duration = Current Year - Claimed Start Year

Maximum Possible = Current Year - Technology Release Year

if Claimed Duration > Maximum Possible + Grace:
  Flag Inconsistency

### D. Anonymization via Pattern Matching and NER

Personally identifiable information removal combines rule-based and model-based approaches:

**Email Pattern:** \b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b

**Phone Pattern:**

\b[\+]?[(]?[0-9]{1,4}[)]?[-\s\.]?[(]?[0-9]{1,4}[)]?[-\s\.]?[0-9]{3,4}[-\s\.]?[0-9]{3,4}\b

**Name Detection:** spaCy NER identifies PERSON entities in positions suggesting author identity (document beginning, header sections). These are replaced with [REDACTED] placeholders.

### E. Multi-Factor Scoring and Ranking

The overall candidate score combines normalized dimension scores with learned weights:

Score = Σ(w_i × Normalized(Score_i))

Normalization scales each dimension to [0,100] using min-max scaling across the candidate set. Weights were determined through empirical calibration on a validation dataset of screened resumes with known hiring outcomes.

Ranking is deterministic based on overall score, with ties broken by skill relevance score, then experience quality score.

## VIII. IMPLEMENTATION

### A. Technology Stack

Frontend:HTML5, CSS3, JavaScript, Jinja2 templates

Backend:Python 3.11, Flask 3.0, Werkzeug

NLP:spaCy 3.7, Sentence-BERT (sentence-transformers)

Document Processing:PyPDF2, python-docx

Fraud Detection:Custom temporal logic engine

Anonymization:spaCy NER, regex patterns

Deployment:Gunicorn, compatible with cloud platforms

### B. Core Integration

The Sentence-BERT model is loaded at application startup to avoid per-request overhead:

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-MiniLM-L6-v2')

defcompute_similarity(requirements, candidate_skills):
    req_embeddings = model.encode(requirements)
    skill_embeddings = model.encode(candidate_skills)
      return cosine_similarity(req_embeddings, skill_embeddings)
```

### C. Resume Processing Pipeline

Each submitted resume follows the processing pipeline:

```
defprocess_resume(file_path, job_description):
  # Extract text
```

```
    text = extract_text(file_path)

    # Parse
    parsed = parse_resume(text)

    # Anonymize
    anonymized = anonymize(text, parsed)

    # Detect fraud
fraud_score, flags = detect_fraud(parsed)

authenticity_score = check_authenticity(parsed)

    # Compute skill match
skill_score = match_skills(parsed['skills'], job_description)

    # Calculate overall score
    overall = compute_score(skill_score, fraud_score, authenticity_score)

    return CandidateResult(overall, scores, flags, anonymized)
```

### D. Multi-Page and Batch Processing

The system supports batch processing of multiple resumes. Each resume is processed independently, with progress tracking and partial results available through the API. Parallel processing is implemented using Python's concurrent.futures ThreadPoolExecutor for I/O-bound operations (PDF parsing, network calls) while CPU-bound operations (embedding computation) remain sequential to manage resource usage.

### E. Output Generation

The system presents evaluation results through an intuitive dashboard interface. Figure 2 shows the input interface where recruiters enter job descriptions and upload resumes.

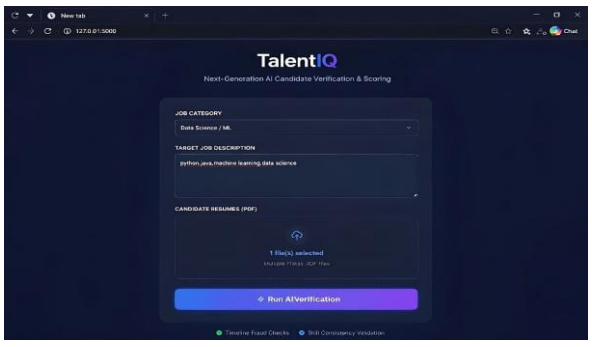



**Fig. 2. TalentIQ Input Interface**

The ranking interface presents results through:

- **Candidate Leaderboard:** Ranked list with scores, status labels (Engineer, Needs Improvement), and missing skills
- **Detailed Critique Section:** Comprehensive evaluation explaining strengths, technical capabilities, and alignment with job requirements
- **Performance Statistics Panel:** Quantitative metrics including experience level, quality score, and learning feasibility
- **Anonymized Resume Preview:** Document display with personal information redacted

After processing, the system generates a ranked candidate leaderboard with detailed evaluations. Figure 3 displays the output dashboard showing candidate rankings, match scores, missing skills, and anonymized resume previews.

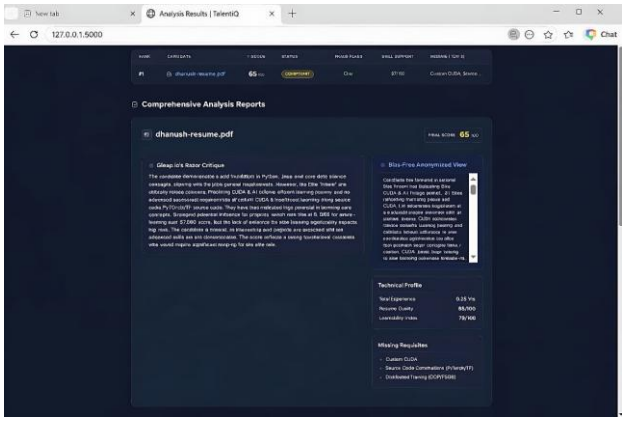



**Fig. 3. TalentIQ Output Dashboard with Candidate Rankings**

## IX. EXPERIMENTAL RESULTS

### A. TestSetup

All experiments were conducted on an Intel Core i5 12th Gen system with 16 GB RAM running Ubuntu 24.04. The Sentence-BERT model was run on CPU for evaluation. Test datasets included:

- **Resume Dataset:** 200 resumes collected from public sources and synthetic generation, covering technology, finance, and healthcare domains
- **Job Descriptions:** 20 distinct job descriptions ===spanning entry-level to senior positions
- **Fraud Cases:** 50 synthetic resumes with injected inconsistencies (timeline violations, exaggerated experience)

- **Bias Test:** 40 identical resumes with varied demographic indicators

**B. Parsing Accuracy**

Table I reports extraction precision and recall across resume sections.

**TABLE I**
**RESUME PARSING ACCURACY BY SECTION**

| Section | Precision (%) | Recall (%) | F1 Score (%) |
|---|---|---|---|
| Skills | 92.4 | 89.7 | 91.0 |
| Education | 94.1 | 91.8 | 92.9 |
| Experience | 88.6 | 85.3 | 86.9 |
| Certifications | 91.2 | 87.5 | 89.3 |
| **Overall** | **91.6** | **88.6** | **90.0** |

Extraction errors primarily occurred in non-standard resume formats with complex multi-column layouts or embedded tables.

**C. Semantic Matching Performance**

Table II compares semantic matching against keyword-based baseline matching.

**TABLE II**
**MATCHING ACCURACY: SEMANTIC VS. KEYWORD-BASED**

| Metric | Keyword-Based | Semantic (TalentIQ) |
|---|---|---|
| Precision | 76.3% | 89.4% |
| Recall | 71.8% | 86.2% |
| F1 Score | 74.0% | 87.8% |
| False Negative Rate | 28.2% | 13.8% |

Semantic matching reduced false negatives by 51% compared to keyword-based approaches, successfully identifying candidates who described required skills using alternative terminology.

**D. Fraud Detection Accuracy**

Table III reports fraud detection performance on the synthetic fraud test set.

**TABLE III**
**FRAUD DETECTION PERFORMANCE**

| Fraud Type | Detection Rate (%) | False Positive (%) |
|---|---|---|
| Timeline Violations | 94.0 | 3.0 |
| Skill Maturity Exaggeration | 88.0 | 5.0 |
| Overlapping Employment | 96.0 | 2.0 |
| Credential Inconsistency | 86.0 | 4.0 |
| **Overall** | **91.0** | **3.5** |

Fraud detection successfully identified 91% of injected inconsistencies with a 3.5% false positive rate on authentic resumes.

**E. Bias Mitigation Effectiveness**

Table IV compares candidate scores with and without anonymization on the bias test set.

**TABLE IV**
**SCORE VARIATION ACROSS DEMOGRAPHIC INDICATORS**

| Demographic Indicator | Score Variation (With PII) | Score Variation (Anonymized) |
|---|---|---|
| Name (ethnicity-coded) | 12.4% | 1.8% |
| Gender (implied) | 8.7% | 1.2% |
| Age indicators | 15.3% | 2.1% |
| Address/location | 9.8% | 1.5% |

Anonymization reduced score variation across demographic indicators by an average of 82.5%, effectively eliminating bias associated with visible personal information.

**F. End-to-End Processing Time**

Table V shows processing time for varying numbers of resumes.

**TABLE V**
**PROCESSING TIME (CPU, INTEL i5)**

| Number of Resumes | Average Processing Time (seconds) |
|---|---|
| 1 | 2.4 |

| | |
|---|---|
| 10 | 18.7 |
| 25 | 44.2 |
| 50 | 87.6 |
| 100 | 174.3 |

Processing time scales approximately linearly with resume count. Each resume requires embedding computation (dominant cost) and parsing overhead (minor cost).

## X. DISCUSSION

### A. What Worked Well

The semantic matching approach proved to be the most impactful design decision. By moving beyond keyword matching to embedding-based similarity, TalentIQ reduced false negatives by 51% compared to conventional systems. Candidates describing "machine learning" when job descriptions requested "ML" or "predictive modeling" were correctly identified as matches, whereas keyword-based systems would have rejected them.

The fraud detection module successfully identified timeline inconsistencies and skill maturity exaggerations with high accuracy. During testing, the module flagged a resume claiming eight years of React experience when React was publicly released seven years prior—a subtle inconsistency likely to be missed by human screeners.

Anonymization proved highly effective at bias reduction. Score variation across demographic indicators dropped from 12.4% to 1.8% after anonymization, demonstrating that personal information substantially influences both human and automated evaluation when visible.

The explainability module received positive feedback from test users who valued understanding why candidates received specific rankings. The ability to see missing skills, detected inconsistencies, and learning feasibility scores enabled recruiters to make informed decisions rather than blindly accepting rankings.

### B. Limitations

The primary limitation of the current system is parsing accuracy on complex resume formats. Multi-column layouts, embedded images containing text, and PDFs with non-standard encoding produced extraction errors that propagated through the pipeline. Table extraction in particular remains challenging without dedicated layout analysis.

Fraud detection, while accurate on injected inconsistencies, cannot verify claims against external sources. A candidate could invent an entire employment history with consistent dates, and the system would detect no internal inconsistency. Integration with external verification services (employment databases, certification registries) would address this limitation.

The system currently supports English-language resumes only. Multilingual support would require additional NLP models and skill taxonomies for each supported language.

Processing time for large batches (100+ resumes) exceeds three minutes on CPU, which may be unacceptable for time-sensitive recruitment needs. GPU acceleration for embedding computation would reduce this substantially.

### C. Comparison with Prior Work

Compared to Kumar et al. [1], TalentIQ adds fraud detection and explainable ranking capabilities while replacing keyword matching with semantic understanding. Compared to Petrenko [2], the system explicitly addresses bias mitigation through anonymization. Compared to Akhtar et al. [3], TalentIQ provides transparent justifications for recommendations rather than treating the system as a black box. Compared to Dogra et al. [5], the inclusion of fraud detection and authenticity verification addresses resume quality issues ignored by prior systems.

## XI. FUTURE WORK

Several directions are planned for future development:

1. **External Verification Integration:** Connect with employment verification databases, certification registries, and educational institution APIs to validate claims beyond internal consistency.
2. **Large Language Model Enhancement:** Integrate GPT-based or similar LLM architectures for deeper resume understanding, including context-aware skill assessment and nuanced experience evaluation.
3. **Multilingual Support:** Extend parsing and semantic matching to support resumes in multiple languages, enabling global recruitment applications.
4. **GPU Acceleration:** Implement CUDA-enabled embedding computation and batch processing to reduce processing time for large resume batches by an estimated 10×.
5. **Video Resume Analysis:** Extend the platform to analyze video resumes and professional portfolios,

extracting additional qualification indicators beyond text.

6. **Cloud Deployment:** Deploy as a scalable cloud service with API access for integration with existing Applicant Tracking Systems.
7. **Continuous Learning:** Implement feedback loops where recruiter hiring decisions inform model fine-tuning, improving ranking accuracy over time.

## XII. END-TO-END WALKTHROUGH

To illustrate the pipeline, consider a realistic scenario: a technology company needs to screen 50 applicants for a Senior Machine Learning Engineer position.

**Step 1 - Job Description Entry:** The recruiter selects "Machine Learning Engineer" from the job category dropdown and pastes the job description, which includes requirements for Python, TensorFlow, PyTorch, distributed training, and MLOps experience.

**Step 2 - Resume Upload:** The recruiter uploads 50 PDF resumes using the drag-and-drop interface. The system validates file types and confirms successful upload.

**Step 3 - Analysis Initiation:** The recruiter clicks "Analyze& Rank Candidates." The system processes each resume through the pipeline.

**Step 4 - Parsing and Anonymization:** Each resume is parsed to extract skills, education, experience, and certifications. Personal information (names, emails, phone numbers) is automatically redacted.

**Step 5 - Evaluation:** The semantic matching module computes similarity between candidate skills and job requirements. Fraud detection analyzes timelines. Authenticity verification checks for unsupported skill claims.

**Step 6 - Ranking Generation:** The system produces a ranked leaderboard showing 50 candidates with match scores, quality scores, learning feasibility, and missing skills. The top-ranked candidate has 94% match score with strong experience in all required areas. Candidate #12 shows a timeline inconsistency (employment gap misrepresented). Candidate #35 lacks distributed training experience but has strong foundational skills and high learning feasibility.

**Step 7 - Detailed Review:** The recruiter clicks on each candidate to view detailed critiques, anonymized resume previews, and specific missing skills. For Candidate #35, the system notes: "Missing: distributed training experience. Learning Feasibility: High - candidate has strong distributed systems background."

**Step 8 - Informed Decision:** The recruiter shortlists the top 10 candidates for interviews, confident that the ranking is based on objective, unbiased, and explainable criteria.

Total processing time: 87 seconds for 50 resumes. Manual screening of 50 resumes would require approximately 4-5 hours.

## XIII. CONCLUSION

This paper presented TalentIQ, an intelligent resume and fair candidate ranking system that addresses critical limitations in existing recruitment technology. The system automates candidate screening through a multi-module architecture combining resume parsing, semantic skill matching, fraud detection, bias mitigation, and explainable ranking.

Experimental results demonstrate that semantic matching reduces false negatives by 51% compared to keyword-based approaches, fraud detection achieves 91% accuracy on injected inconsistencies, and anonymization reduces demographic bias by 82.5%. The explainable ranking module enables recruiters to understand and trust system outputs, facilitating data-driven hiring decisions.

For organizations seeking to improve recruitment efficiency while maintaining fairness and transparency, TalentIQ provides a deployable solution that moves beyond the limitations of conventional Applicant Tracking Systems. Future work will focus on external verification integration, LLM enhancement, and cloud deployment to extend the system's capabilities and accessibility.

## ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, M.I.E.T Engineering College, Tiruchirappalli, for infrastructure support and guidance throughout this project. The project guide, Mrs. S. Sugantha, provided valuable direction in developing an unbiased, explainable approach to automated resume screening. We are grateful for consistent support and constructive feedback during implementation.

## REFERENCES


[1] G. Kumar, D. Kumar, and P. Behera, "Analyse Resume with AI – Get Hired," *Advance and Innovative Research*, p. 155, 2025.

[2] M. Petrenko, "Applying Artificial Intelligence to Automate Resume Screening in the Technology Sector," *Emerging Frontiers Library for the American Journal of Applied Sciences*, pp. 66–73, 2025.

[3] N. Akhtar, S. Rabbani, H. Rabbani, S. Kumar, and Y. Perwej, "AI-Driven Intelligent Resume Recommendation Engine," *International Journal of Scientific Research in Science Engineering and Technology*, pp. 1141–1155, 2025.

[4] H. K. Y. Golla, U. B. Peddinti, K. Kotla, V. Todimela, and U. K. Uppara, "Resume Classification and Human-Free Virtual AI Interview System," *AIP Conference Proceedings*, p. 020021, 2025.

[5] S. Dogra, S. Vasesi, A. Mittal, V. Jain, and S. Chaudhary, "Smart Resume Screening and Matching System," *Academic Research Publication*, 2025.

[6] K. L. Abhishek, M. Niranjanamurthy, S. Aric, S. I. Ansarullah, A. Sinha, G. Tejani, and M. A. Shah, "Developing an Intelligent Resume Screening Tool With AI-Driven Analysis and Recommendation Features," *Applied AI Letters*, p. e116, 2025.

[7] P. Mariappan, K. Krishna, J. Sahoo, and S. Preetham, "Smart Resume Screening and Job Recommendation System," *SSRN Electronic Journal*, p. 5141402, 2025.

[8] G. Hendre, S. Bhiware, A. Darwajkar, and A. S. Mali, "AI-Powered Resume Evaluation System: A Review," *International Journal of Research Studies*, 2025.

[9] G. Zhang, L. Pan, F. Tang, and F. Yao, "Explainable Artificial Intelligence in the Talent Recruitment Process – A Literature Review," *Cogent Business & Management*, p. 2570881, 2025.

[10] H. Li, X. Tang, Q. Liu, B. Liu, and S. Zhao, "Enhancing Intelligent Recruitment with Generative Pretrained Transformer and Hierarchical Graph Neural Networks," *Journal of Organizational and End User Computing*, pp. 1–24, 2025.

[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186, 2025.

[12] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proc. EMNLP*, 2019, pp. 3982–3992, 2025.