ResumeMatch: A Cloud-Native SaaS Platform for Semantic Resume-Job Alignment

Kabir Roy
Department of Computer Science & Engineering

Abstract
We present ResumeMatch, an open-source cloud-native SaaS application that applies established Natural Language Processing techniques to the resume screening problem. Unlike traditional Applicant Tracking Systems that rely on rigid keyword matching, ResumeMatch implements a hybrid comparison engine utilizing Sentence-BERT for semantic similarity and explicit skill extraction for interpretability. The system integrates a Generative AI feedback mechanism to provide candidates with actionable resume optimization strategies. This paper describes the system architecture, implementation decisions, and performance characteristics of the platform, demonstrating its viability as a real-time tool for democratizing access to advanced NLP in career technology.

Keywords: *Applied NLP, System Design, Sentence Transformers, Recruitment Technology, Open Source*

1. Introduction

The mismatch between candidate qualifications and job description phrasing represents a persistent challenge in automated recruitment. While transformer-based matching has been extensively studied [1, 2], few accessible, production-ready tools implement these techniques in a user-facing format suitable for individual job seekers.

ResumeMatch addresses this implementation gap by providing an open-source, cloud-deployable platform that makes State-of-the-Art NLP accessible beyond enterprise Applicant Tracking Systems. Our contribution is not a novel algorithm, but rather a thoughtful system integration that balances semantic understanding with interpretability, wrapped in a production-ready architecture.

1.1 Design Goals

Semantic Matching: Move beyond keyword overlap to contextual understanding
Explainability: Provide transparent scoring with identified skill gaps
Actionability: Generate concrete improvement suggestions via LLMs
Accessibility: Free, open-source tool for job seekers
Performance: Real-time response suitable for interactive use

2. System Architecture

ResumeMatch is built as a decoupled microservices architecture to ensure scalability, maintainability, and cloud portability.

2.1 Technology Stack

Frontend Layer:
- React.js with Vite bundler for optimal load performance
- Custom CSS with neon-themed UI for visual distinction
- Responsive design for mobile and desktop

Backend Layer:
- FastAPI (Python 3.10+) for high-performance async request handling
- Uvicorn ASGI server with WebSocket support for streaming
- RESTful API design with OpenAPI documentation

NLP Pipeline:
- sentence-transformers library for SBERT inference

- Model: all-MiniLM-L6-v2 (80MB, 384-dimensional embeddings)
- pdfminer.six for PDF text extraction with layout awareness

Generative Layer:
- Groq API integration for Llama-3-8B-Instant inference
- Structured JSON output prompting for resume recommendations
- Fallback to local generation if API unavailable

Deployment:
- Frontend: Vercel (edge deployment)
- Backend: Render (containerized Python service)
- GitHub Actions CI/CD for automated testing

2.2 Processing Pipeline

The system processes resume-JD pairs through a four-stage pipeline:

Stage 1: Document Parsing
Input: PDF Resume → pdfminer.six → Plain Text
- Preserves formatting for contact extraction
- Handles multi-column layouts
- Filters headers/footers

Stage 2: Skill Extraction
Explicit Skills: Regex patterns for technologies
- Programming languages (Python, Java, C++, etc.)
- Frameworks (React, Django, TensorFlow, etc.)
- Tools (Git, Docker, AWS, etc.)
Output: Set of detected skills with frequency counts

Stage 3: Semantic Embedding
SBERT Encoding:
Resume Text → all-MiniLM-L6-v2 → V_r (384-dim vector)
JD Text → all-MiniLM-L6-v2 → V_jd (384-dim vector)
Cosine Similarity: sim = (V_r · V_jd) / (||V_r|| ||V_jd||)

Stage 4: Hybrid Scoring
S_final = α · S_skills + β · S_semantic
Where:
- S_skills = Jaccard(skills_resume, skills_jd)
- S_semantic = cosine_similarity(V_r, V_jd)
- α = 0.7, β = 0.3 (tuned empirically for interpretability)

2.3 Generative Feedback Engine

When a match score is below threshold or upon user request, the system invokes the Llama-3 model via Groq's inference API with a structured prompt:
Given:
Resume: {resume_text}
Job Description: {jd_text}
Detected Skills: {resume_skills}
Required Skills: {jd_skills}

Generate JSON:

{ "summary": "...", "skills_to_add": [...], "bullet_improvements": [...] }

The structured output ensures reliability and allows direct UI rendering without additional parsing.

3. Implementation Details

3.1 Performance Optimization

Cold Start Mitigation:
- SBERT model loaded once at server initialization
- Kept in memory across requests (persistent workers)
- Warm-up request during deployment for instant first response

Inference Optimization:
- Batch processing for multiple resume comparisons
- Cached embeddings for frequently-used JD templates
- Async processing for non-blocking generative feedback

Resource Management:
- Model runs on CPU (80MB RAM footprint)
- Suitable for free-tier cloud hosting (512MB instances)
- No GPU required for inference at this scale

3.2 Latency Benchmarks

Testing environment: Render Free Tier (512MB RAM, shared CPU)

Operation	Latency	Notes
PDF Parsing	45-120ms	Varies with document complexity
SBERT Inference	12-18ms	Per resume-JD pair
Skill Extraction	3-8ms	Regex-based, deterministic
Hybrid Scoring	<1ms	Simple arithmetic
Groq API Call	800-1500ms	Network + generation time
Total (without AI)	~180ms	Interactive performance
Total (with AI)	~2s	Acceptable for async UX

These measurements demonstrate that the SBERT architecture adds minimal overhead (<20ms) while providing significant semantic capability, making it highly suitable for real-time applications.

3.3 User Experience Design

Progressive Disclosure:
- Immediate visual feedback during upload
- Parsed resume preview with detected information
- Score computation with animated progress
- Expandable sections for detailed breakdowns
- Optional AI suggestions on demand

Explainability Features:
- Color-coded skill badges (matched vs. missing vs. bonus)
- Visual match score gauge with percentage
- Detailed breakdown: skill match % and JD similarity %

- Specific improvement recommendations with rationale

4. Discussion

4.1 System Contributions

1. Democratized NLP Access
ResumeMatch brings transformer-based semantic matching—typically reserved for enterprise ATS platforms—to individual job seekers as a free, open-source tool. The entire system runs on free-tier cloud infrastructure, ensuring accessibility.
2. Hybrid Interpretability
By combining explicit skill extraction (α=0.7) with semantic similarity (β=0.3), the system provides both precise skill gap analysis and holistic context matching. This addresses the "black box" criticism of pure neural approaches.
3. Constructive Feedback Loop
Rather than a binary accept/reject signal, the generative layer provides actionable improvement strategies. This shifts the paradigm from "screening" to "coaching."
4. Open Source Implementation
Complete source code, deployment configurations, and documentation are publicly available, enabling reproducibility and community contributions.

4.2 Limitations and Future Work

Current Limitations:
- Resume parsing assumes standard formats; creative layouts may fail
- Skill extraction uses fixed regex patterns (English-centric)
- No multi-language support
- Scoring weights (α, β) are manually tuned, not learned
- No personalization or user history tracking

Planned Improvements:
- Fine-tune SBERT on domain-specific resume-JD pairs
- Add support for experience level weighting (junior vs. senior)
- Implement A/B testing framework for scoring algorithms
- Build anonymized user feedback collection for model improvement
- Expand skill taxonomy beyond software engineering

4.3 Ethical Considerations

Privacy: All processing happens server-side with no persistent storage of resume content. Users can optionally save results locally.
Bias Mitigation: Semantic models can inherit biases from training data. Future work will include fairness audits and bias detection mechanisms.
Transparency: Open-source nature allows scrutiny of algorithms and prevents proprietary "black box" gatekeeping in hiring.

5. Related Work

Resume-Job Matching:
Maheshwari et al. [1] demonstrated BERT's effectiveness for resume screening. Our work extends this by adding hybrid scoring and generative feedback in a production system.
Sentence Transformers:
Reimers & Gurevych [2] introduced SBERT for efficient semantic similarity. We apply this specifically to career documents with domain-specific weighting.
Explainable AI in HR:
Recent work emphasizes interpretability in hiring algorithms [3]. Our hybrid approach directly addresses this by combining symbolic (skill matching) and neural (SBERT) methods.

6. Conclusion

We presented ResumeMatch, a production-ready SaaS platform that demonstrates the practical application of established NLP techniques to career technology. By thoughtfully integrating SBERT embeddings, explicit skill extraction, and generative AI feedback, we provide a transparent, actionable, and accessible tool for job seekers.

The system is open source and deployed at [GitHub repository link], with live demo available. We invite the research community to use, critique, and extend this work as a foundation for further innovation in recruitment technology.

Code Availability: https://github.com/Kabirroy12345/resume-match-engine

References

[1] S. Maheshwari, S. Sajnani, and A. Garg, "Resume Screening using Bidirectional Encoder Representations from Transformers (BERT)," 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2020.

[2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL-HLT, 2019.

[4] A. Raghavan, S. Barocas, K. Levy, and S. Narayanan, "Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices," Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 2020.

[5] A. Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems 30 (NIPS), 2017.