FMDB *Transactions on Sustainable Computing Systems*

Resume AI: An Explainable Hybrid Agentic Resume Ranking System Using Local LLMs and Deterministic NLP

Aashutosh Indusekhar[1,*], R. Manikandan[2], T. Sanjay Karthik[3], V. Nivedita[4], Targyn A. Nauryz[5], S. J. Vimal Aravintha[6]

[1,2,3,4]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.
[5]School of Digital Technologies, Narxoz University, Almaty, Kazakhstan.
[6]Department of Data Science, Indiana University Bloomington, Bloomington, Indiana, United States of America.
ai9319@srmist.edu.in[1], mr6539@srmist.edu.in[2], st9076@srmist.edu.in[3], niveditv1@srmist.edu.in[4], t.nauryz@kbtu.kz[5], vimasubr@iu.edu[6]

Abstract: The large number of applications that recruiters need to go through has made the limitations of traditional Applicant Tracking Systems (ATS) increasingly hard to ignore. The keyword-matching logic, lack of transparency in scoring, and vulnerability to resume cheating are issues that traditional systems have not addressed. This paper presents Resume AI, a hybrid agentic resume ranking system that uses a two-layer architecture. The first layer is a deterministic NLP engine that deals with all scoring tasks using a four-component weighted formula: Skill-set intersection, TF-IDF job description similarity, experience weighting, and a bounded LLM decision score. The second layer is the locally running large language model that interprets the structured resume and structured job description, makes candidate ranking decisions with a capped contribution of 20% to the overall score, and generates recruiter explanations. All this inference happens on the device with Ollama, with no data sent to any service. Tested with the keyword-only baseline on 15 resume-JD pairs, the system can correctly penalise modified resumes, handle semantic synonyms that exact matching would miss, and generate explanations based entirely on deterministic data for recruiters.

Keywords: Resume Parsing, Explainable AI, Hybrid AI Architecture, Agentic LLM, Job Description Matching, Skill Gap Analysis, Candidate Ranking, Recruitment Intelligence, Local LLM, Weighted Scoring, LLM Decision Agent.

Received on: 16/05/2025, **Revised on:** 11/07/2025, **Accepted on:** 12/10/2025, **Published on:** 12/06/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCS

DOI: https://doi.org/10.69888/FTSCS.2026.000688

Cite as: A. Indusekhar, R. Manikandan, T. S. Karthik, V. Nivedita, T. A. Nauryz, and S. J. V. Aravintha, "Resume AI: An Explainable Hybrid Agentic Resume Ranking System Using Local LLMs and Deterministic NLP," *FMDB Transactions on Sustainable Computing Systems,* vol. 4, no. 2, pp. 149–164, 2026.

1. Introduction

Recruitment at scale is as much a coordination problem as it is an evaluation problem [1]. When an organisation receives hundreds of applications for a single job, the problem is no longer a lack of applicants, but the inability to devote each application the consideration it deserves within a reasonable time. This problem was solved by Applicant Tracking Systems,

*Corresponding author.

which have done so at the expense of the evaluation process, reducing it to keyword counting in the most widely used platforms. The implications of this are well understood [7]. A candidate with actual React skills who talks about "frontend component libraries" will be filtered out by an ATS searching for the keyword "React" [2]. A candidate with no relevant skills who copies and pastes the job description will be ranked highly. Neither candidate would be ranked as such by a recruiter who carefully read the same applications [3]. To add insult to injury, these systems provide rankings without explanation, leaving a recruiter to interpret a list with no explanation, candidates receive rejections without a rationale, and organisations gain no insights from the process. Regulatory advancements have heightened these issues [13]. GDPR Article 22, in effect since 2018, grants data subjects the right to object to decisions made by fully or partially automated decision-making processes that have significant effects on them [4]. Article 22 requires that decision-making processes, including AI, justify the decision when requested [14].

The proposed AI Act in the European Union, which regulates AI, considers AI-based candidate selection tools to be high-risk AI that requires transparency and accountability [15]. Most ATS tools used in practice cannot fulfil these requirements. A third issue is that organisations may not have considered using LLMs in their candidate selection process. With cloud-based LLMs, data transfer requirements pose challenges for GDPR compliance. Resume data, in fact, is considered personal data under Article 4. Most seriously, there is a risk of hallucination, or the LLM generating claims about a candidate that are not supported by the candidate's actual documents. Resume AI mitigates all three failure modes with a hybrid approach that maintains a deterministic scoring model and local explainability. A four-component weighted formula is used to make all ranking choices. A locally executed language model is used for structured data extraction and, crucially, provides a constrained 20% decision signal to the result via a reasoning call that serves as a decision-support agent rather than a free-form generator. A fourth inference call generates a natural-language explanation for the recruiter based solely on the scoring layer's outputs. The contributions of this work can be listed as follows:

- A revised scoring formula with four components, which includes a bounded LLM decision score with 20% weight, deterministic skill matching with 40% weight, TF-IDF similarity with 25% weight, and experience weighing with 15% weight, with the LLM contribution capped and anti-hallucination-constrained.
- A four-call local LLM inference pipeline for candidate resumes NLU, job description NLU, bounded decision scoring, and recruiter explanation generation, each with output constraints.
- A fully local deployment architecture with Ollama, which processes all resume data on-device, thus satisfying GDPR data residency requirements without additional compliance overhead.
- Experimental validation of the approach, showing that it correctly penalises keyword-stuffed resumes, has improved handling of semantic synonyms, and provides recruiter explanations per candidate, which were not available in any of the baseline approaches that used only keywords.

1.1. Problem Statement

The underlying problem in the automated screening of resumes is that the three attributes most desirable to the recruiter, semantic understanding, transparent reasoning, and defensible ranking, are exactly the attributes that are not provided by keyword-based ATS systems. In the context of semantic understanding, it is recognised that "distributed systems experience" and "Kubernetes orchestration background" are equivalent for a specific job description [5]. It is not possible to have a list of keywords that can cover all such equivalences, and it is not feasible to maintain such a list given the rate at which technology changes. Machine learning approaches fare better, but it is not possible to explain the results, as it is not possible to explain what cosine similarity between two vectors means to a recruiter. Transparent reasoning is a necessity in some countries and in any hiring process that candidates or regulators can review. GDPR Article 22 requires explanation for decisions made by automated processes that affect individuals. The proposed EU AI Act regulates AI in employment and requires documentation of decision logic, which current ATS providers do not support, but the regulation demands. A defensible ranking means that if a recruiter is asked to explain why Candidate A was selected, and Candidate B was not, the system should have a concrete answer tied to the actual candidate documents. Black-Box Scoring, whether performed via keyword analysis or neural networks, cannot do this. The research question this work answers is: Can some system rank candidates with semantic awareness comparable to that of a language model, and can those rankings be fully auditable, explainable to a recruiter, and computed entirely on local hardware?

1.2. Research Objectives

The system is meant to achieve the following objectives:

- Develop a hybrid architecture in which scoring and explanation are separated: The scoring portion must be deterministic and reproducible; the explanation portion must be grounded in the outputs of the scoring portion and must not make claims not grounded in those outputs.

- Develop a curated skill database and a deterministic skill extraction engine that will identify matched skills and unmatched skills for any given pair of resume and JD, with identical results for repeated execution.
- Add a bounded LLM decision component that contributes to the overall candidate score of up to 20% via a constrained reasoning call.
- Develop a four-call local inference pipeline per candidate for structured resume parsing, structured JD parsing, bounded decision scoring, and natural language explanation generation.
- Execute the full inference pipeline on local hardware via Ollama, preventing any resume content from being sent to external services.
- Produce structured recruiter outputs, ranked lists, skill gap reports, domain profiles, and natural language explanations based solely on auditable data sources.

2. Literature Survey

The literature on automated resume screening has covered rule-based approaches, traditional machine learning, deep learning, and, more recently, the integration of large language models. The following literature survey categorises existing work by methodology and outlines the problem addressed by this system.

2.1. Keyword-Based and Rule-Based Approaches

The first resume screening systems and the most current commercial ATS solutions are based on keyword lists extracted from job postings. Chifu et al. [5] introduced a skill extraction system based on rule-based pattern matching and part-of-speech tagging for structured resume formats. While the system performed well on resumes with standard layouts, it performed poorly on resumes with non-standard section titles or multi-column layouts. This limitation is practical, given the variability of resume design choices in real-world applications. Nawaz and Gomes [4] investigated the role of AI-enhanced chatbots in hiring and found that automation enhanced efficiency but did not enhance fairness or transparency. Another study by Nawaz [6] on robotic process automation in hiring concluded that keyword pipelines are prone to perpetuating biases rather than mitigating them, as keyword lists are based on past job postings that may also be biased by hiring criteria.

2.2. Machine Learning for Resume Analysis

Machine learning techniques were incorporated to overcome the brittleness of rule-based matching. Gugnani and Misra [8] proposed implicit skill extraction via document embeddings for job suggestions. They showed that distributed representations of skill context are more effective than keyword matching for skills described in paper descriptions rather than in skill lists. Named Entity Recognition models based on Conditional Random Fields and Support Vector Machines have been used for structured information extraction from resumes. These models improve extraction accuracy on labelled data but require large amounts of annotated data and still fail to provide explanations for their ranking results. TF-IDF and BM25 models have been used for resume-JD matching as a means of vocabulary-level semantic representation; the current system includes TF-IDF similarity in its ranking formula for this purpose.

2.3. Deep Learning and Transformer-Based Approaches

Recurrent networks (RNNs), LSTM networks, and convolutional networks (CNNs) have been used for resume parsing, enabling better feature extraction in context than traditional ML methods. BERT, developed by Devlin et al. [9], was a breakthrough: Its bidirectional contextual understanding greatly improved semantic similarity matching for skill synonym identification and role assignment. BERT-based dense embedding techniques have since become the norm in research-level resume similarity matching systems. Ashrafi et al. [10] developed Career-gAIde, a CNN-powered career recommendation system that achieved 67% precision and 84% accuracy in resume skill extraction. The system was highly effective at predicting career paths. Still, it was cloud-based and offered no way to explain candidate rankings to hiring managers-a common issue with deep learning models in this area. A common weakness of transformer and embedding models is that their similarity scores are not interpretable in recruiter language. A vector distance cannot be expressed as "this candidate lacks Kubernetes skills." Resume AI mitigates this issue by ensuring deterministic scoring and leaving natural language explanation to a local language model.

2.4. Explainability and Large Language Models in Recruitment

Explainable AI research has shown that explanation mechanisms increase user trust in AI-supported decisions [11]. In the context of employment, this is more than a usability issue. GDPR Article 22 and the EU AI Act require explanation for each decision as a matter of compliance for consequential AI outcomes. Post-hoc XAI methods like LIME and SHAP offer feature

attribution explanations for neural networks but are technically interpretive and not designed for communication with recruiter-facing users. Large language models offer another path to explanation: they can take structured scoring data as input and generate free-text explanations that users can read and respond to Miller [12]. However, the danger is that LLMs, left to their own devices to generate text about a candidate, may generate claims not justified by the candidate's documents. This risk has not been systematically mitigated in prior work on LLM-based recruitment systems. The architecture described below mitigates this risk through output-constrained prompting, temperature-zero inference, and strict isolation of the explanation call from the scoring pipeline. The current work also extends this architecture by incorporating a bounded LLM decision call, a structured inference component that allows the model to assess the fit of candidates based on candidate profiles extracted by NLU and aligned with the JD requirements, and returns a normalised score between [0, 1]. The returned score accounts for 20% of the overall ranking, providing semantic judgment capability, while 80% of the ranking is under deterministic control.

2.5. Research Gap

The current automated resume screening systems are deficient concerning one or more of the following:

- Exclusive dependence on surface-level matching
- No provision for an explanation mechanism
- Dependency on cloud infrastructure for LLM inference
- Unbounded LLM participation in scoring decisions

Resume AI overcomes all four of these limitations simultaneously, as no current system in the literature has been identified that achieves this combination of capabilities.

3. Proposed System Architecture

3.1. System Overview

The system runs on a two-layered structure. The Deterministic NLP Engine is responsible for text extraction, skill identification, contact profiling, and TF-IDF scoring operations. These operations always produce the same output for the same input. LLM Inference Layer takes the output from the Deterministic NLP Engine. It performs four operations: Interpreting a structured resume, interpreting a structured job description, candidate ranking decision scoring (capped at 20% of the final score), and generating recruiter explanations. Figure 1 illustrates this architecture.

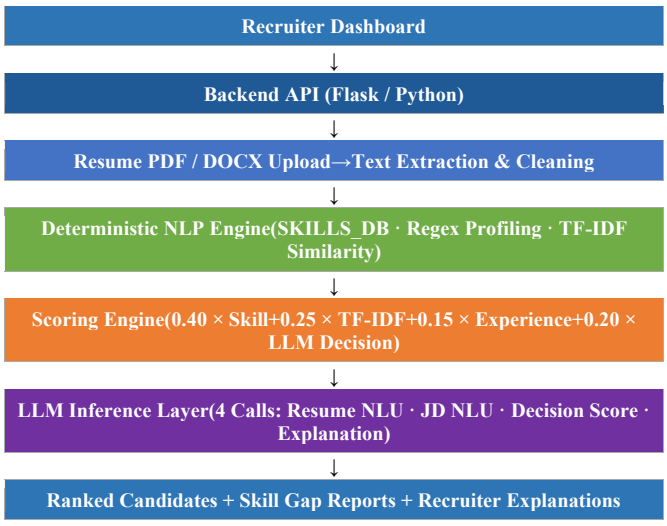

Figure 1: System architecture and data flow

Figure 1 presents the complete architecture of the proposed system, with its seven layers of functionality. Reading Figure 1 from top to bottom shows the progression of the recruitment session from input to output, with each layer passing its specifically generated output to the next. No layer passes on its output back up the stack, and no layer below the scoring engine has any influence on what the layers above it have generated. It is this unidirectional data flow that enables every ranking decision to be independently verifiable. The recruitment session begins with the Recruiter Dashboard, which serves as the sole entry point

for resume documents and job descriptions. The recruiter sends one or more resume documents (PDF or DOCX) along with the job description (free text). This is handled by the Recruiter Dashboard, which sends it to the Backend API. The Backend API handles everything, serving as the coordinator rather than the processor, ensuring everything remains independent and self-testable. From the Backend API, uploaded documents are sent to the Text Extraction and Cleaning phase. This phase addresses the different formats in which actual resume submissions come. There can be PDF files that have been represented as searchable text, PDF files represented as images, DOCX files that have multi-column formatting, DOCX files that have been represented as having tables within them, and documents that have been represented in any or all of the above formats.

This process normalises whitespace, strips non-semantic special characters, performs Unicode normalisation, and relies upon typographic and keyword-based detection to recognise section breaks. The output of this process will be clean text marked for sections, an exact representation that doesn't rely on the original document type. This now goes into the Deterministic NLP Engine. This is where all our rule-based intelligence lives. Candidate technical skills matching is performed using SKILLS_DB, contact information regex-based field extraction for creating candidate profile headers, domain category scoring across six skill domains, and TF-IDF cosine similarity between resume text and job description text. All operations in this layer are stateless and deterministic, meaning that if researchers take a resume and a SKILLS_DB, they can run this operation as many times as they want. The result will always be the same. That is where our auditability claim comes in. The Scoring Engine takes the outputs of the NLP layer and combines them into a final score using the four-part weighted equation: 0.40 × Skill Match Score, 0.25 × JD Similarity Score, 0.15 × Experience Weight, and 0.20 × LLM Decision Score. Three of the four inputs come from the deterministic layer above. The fourth, the LLM Decision Score, is calculated in the inference layer below and returned as a bounded scalar. The scoring engine is not an inference engine; it is an arithmetic operation on numbers that other layers have already settled on. This means that a recruiter or auditor can calculate the final score accurately by looking up the four parts of the equation and crunching the numbers, without running the model.

The LLM Inference Layer is the layer at the base of the pipeline that handles the four calls the pipeline makes to the model for each candidate. Resume NLU and JD NLU are the first two calls, in which the model returns structured JSON objects for the resume and JD text, respectively. These objects then go into the decision scoring call, which is the third call. In this call, the model receives the two objects and returns a float in the range [0,1] representing its assessment of the candidate. This float is the LLM Decision Score. Explanation Generation, the fourth call, is provided with all the scoring outputs and is expected to generate a recruiter-readable paragraph based solely on those outputs. What's important to notice, however, is that the LLM Inference Layer cannot write back to any layer above the scoring engine. It takes inputs from the NLP engine outputs and produces outputs to inform the scoring engine and the dashboard, but it cannot change what the deterministic layers have already determined. The final layer – Ranked Candidates, Skill Gap Reports, and Recruiter Explanations – is what the recruiter views. Each candidate in the result set has a final weighted score, a verdict label, a list of matched skills, a list of missing skills, a structured NLU-extracted profile, an LLM decision score with its reasoning trace, and a natural language explanation paragraph. This result set is much more detailed than the shortlist or reject verdicts that a traditional ATS delivers, and all the information in it traces back to a named computation step in the layers above.

3.2. Resume Parsing and Text Extraction

The parsing module is the first active component a resume encounters after it is uploaded, and it has a deceptively difficult task. Resume files come in two formats, PDF and DOCX, but the variability in structure within each format is staggering. A PDF generated from exporting a Word document has selectable, searchable text that can be programmatically extracted at the paragraph level. A PDF generated by scanning a physical resume and saving it as an image contains no extractable text, as it lacks optical character recognition. Between these two extremes are PDFs generated from design software, online resume builders, and LaTeX templates, each with its own set of internal encoding oddities. The extraction module handles all these varieties through format detection and format-specific extraction logic. With DOCX files, the extraction process poses a different challenge. DOCX files contain XML files in a ZIP archive, and the text within the document is encoded to differentiate paragraph runs, Table cells, text boxes, headers, and footers. A simple text reader that reads text line by line will not only fail to read text from Table cells but will also read it in a manner that is not in the order that a human reader will read it. The extraction module reads the XML structure explicitly to read paragraph runs and Table cell text in the correct order. This is important when parsing resumes that use tables to create invisible grids for multi-column layouts, a common technique. In such cases, the actual list of skills or work history may be encoded within Table cells.

In addition to format-specific extraction, the module must also handle the deliberate formatting variations introduced by candidates. Multi-column formatting puts information side by side on the page, but the formatting varies depending on the tool used. Graphic skill bars, rating stars, and skill rings are purely graphical – they encode no extractable text and are ignored in any text-based parse. Section headings are completely unstandardized – what one candidate labels as Professional Experience might be labelled Work History, Career Summary, Employment Record, or omitted in favour of an unmarked date range by another. The extraction module uses a combination of whitespace normalisation, Unicode normalisation, special character

stripping, and typographic pattern recognition – capitalisation, font size markers where available, and keyword proximity – to determine section boundaries and assign each text area its likely section label. This section's labelling is what enables the other components to distinguish work history text from education text and from skills section text. A quality signal runs alongside extraction. The module calculates the percentage of printable characters to the amount of whitespace characters in the extracted text. Legitimate text-heavy resumes will have high percentages; image-only PDFs and heavily formatted resumes that extract very little text will have percentages below some threshold we've learned over time. When researchers detect this flag, they continue processing the resume, even though they know the extraction quality was poor, because some data is better than none.

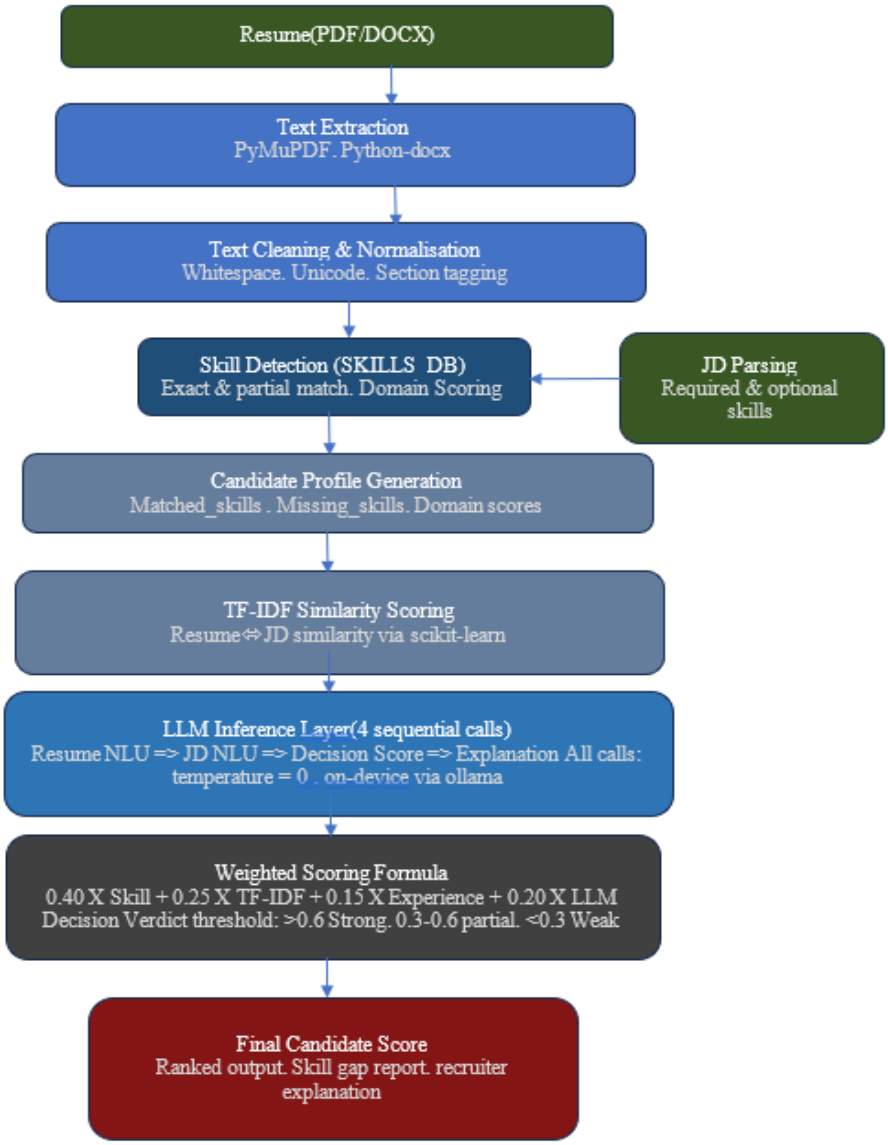

Figure 2: Resume AI data flow pipeline

Researchers note this extraction quality flag in the candidate's output profile. Hence, the recruiter understands that the score may not be based on the full contents of the resume, but only on the content researchers were able to extract. Researchers make this transparency choice because they would never drop a candidate based on an extraction problem, and they would never represent the score as highly accurate when they know the text they have is of poor quality. Figure 2 displays the complete lifecycle of the resume document from raw file upload to final scored output. What this diagram makes visible, however, is easy to underestimate at a high level: The sheer number of transformation steps between when the recruiter makes the upload request and when the scored output appears on the screen. Each node in Figure 2 represents a single step in the pipeline, with a well-defined task. Text Extraction makes no scoring decisions. Text Cleaning knows nothing of the skills required within the job description. Skill Detection knows nothing of the weight; the skill match score will be given. Candidate Profile Generation

makes no calls to LLM. TF-IDF Similarity Scoring does not have access to the candidate's project descriptions. It treats both documents as bags of weighted terms. The scores produced by the deterministic nodes in the layers above it remain unchanged by the LLM Inference Layer. The strict separation between the stages just described is what makes the process auditable. A recruiter, a compliance officer, or an auditor can look at any single node and know that it performed only its designated function: nothing.

3.3. Deterministic NLP Engine and SKILLS_DB

The NLP engine is structured around SKILLS_DB, a carefully constructed hierarchical taxonomy of technical skills that is divided into six skill domains: Programming Languages, Web Frameworks, Cloud and Infrastructure, DevOps and Containerization, Data and Machine Learning, and Database Technologies. The current taxonomy includes 29 skills and is intended to be easily extensible by HR administrators without requiring code modifications:

```
SKILLS_DB = {
"programming": ["Python", "Java", "JavaScript", "C++"],
"web": ["React", "Node.js", "Flask", "FastAPI"],
"cloud": ["AWS", "GCP", "Azure"],
"devops":["Docker", "Kubernetes", "CI/CD"],
"data_ml": ["Machine Learning", "Deep Learning", "NLP"],
"databases": ["SQL", "MongoDB", "PostgreSQL"]
}
```

Extraction runs case-insensitive exact and partial matching against the cleaned resume text. Each detected skill is tagged with its source category. The extraction produces three per-candidate structures: Candidate_skills (all skills found), matched_skills (the intersection with JD-required skills), and missing_skills (JD-required skills absent from the candidate's profile). Because the logic is purely rule-based, the output is guaranteed reproducible across runs. A separate regex layer extracts contact fields: Candidate name (the first line of four words or fewer within the document's opening five lines), email address, and phone number, including an optional country code. Domain scoring aggregates skill counts per category into a six-dimensional profile used by the explanation layer for domain characterisation.

3.4. Scoring Engine and Four-Component Formula

The final candidate score is calculated by combining four components. The updated formula is:

```
Final Score =
0.40 × Skill Match Score
+ 0.25 × JD Similarity Score
+ 0.15 × Experience Weight
+ 0.20 × LLM Decision Score
```

The Skill Match Score is the fraction of JD-required skills found in the candidate's resume. It is calculated as |candidate_skills ∩ jd_required_skills| / |jd_required_skills|. This is a simple set intersection calculation, and the result is determined solely by the SKILLS_DB extraction:

- **JD Similarity Score:** The JD Similarity Score is the similarity in language between the entire resume text and the entire JD text. TF-IDF vectors are computed for each text and then used to compute cosine similarity. This score measures contextual language matches in paper descriptions, work history, and job summary text that are completely ignored in the keyword-based approach.
- **Experience Weight:** Years of experience extracted by the LLM are mapped to a value in the range [0, 1] using a piecewise linear function. Zero years of experience maps to 0.0, 2 years of experience maps to 0.5, and 5 or more years of experience maps to 1.0. Note that this is an integer, and the LLM calls the resume NLU to get this, but it does not determine the weight.

The LLM Decision Score is the new metric introduced in this study. A dedicated inference call is made to obtain the candidate's structured NLU profile and the JD's structured NLU profile, and a float in the range [0, 1] representing the candidate's fit, along with a short reasoning trace, is requested. The score is clamped to [0, 1], and if it is outside that range, it is clamped to the nearest endpoint. Furthermore, information not in the profiles is not to be mentioned. This 20% weight is giving the model a say in the rankings, but not enough to overrule the deterministic majority. The rankings are sorted by Final Score in descending order; in case of a tie, they are sorted by Skill Match Score, then by experience, and finally by document name. Score thresholds

determine the verdicts: Strong Fit if the score is greater than 0.6, Partial Fit if the score is in the range (0.3, 0.6], and Weak Fit if the score is less than 0.3 (Table 1).

Table 1: Scoring component breakdown

Component	Weight	Computation Method	Data Source	Hallucination Risk
Skill Match Score	40%	Set intersection ratio	SKILLS_DB (deterministic)	None
JD Similarity Score	25%	TF-IDF cosine similarity	Raw text (deterministic)	None
Experience Weight	15%	Piecewise linear on extracted int	LLM NLU (integer only)	Minimal
LLM Decision Score	20%	Constrained LLM reasoning call	LLM (bounded [0,1])	Mitigated by capping and output constraints

Figure 3 illustrates the convergence of the four inputs into the final score. Three of the four inputs come from rule-based, deterministic operations, whose outputs will always be the same for a given pair of resume and JD, regardless of how many times the system runs. The fourth input, the LLM Decision component, injects a bounded semantic signal. It can move the score, but cannot override a decision that the deterministic 80% output of the other three components has already made. The NEW designation for this component highlights it as the key innovation of this paper over the prior three-component models.

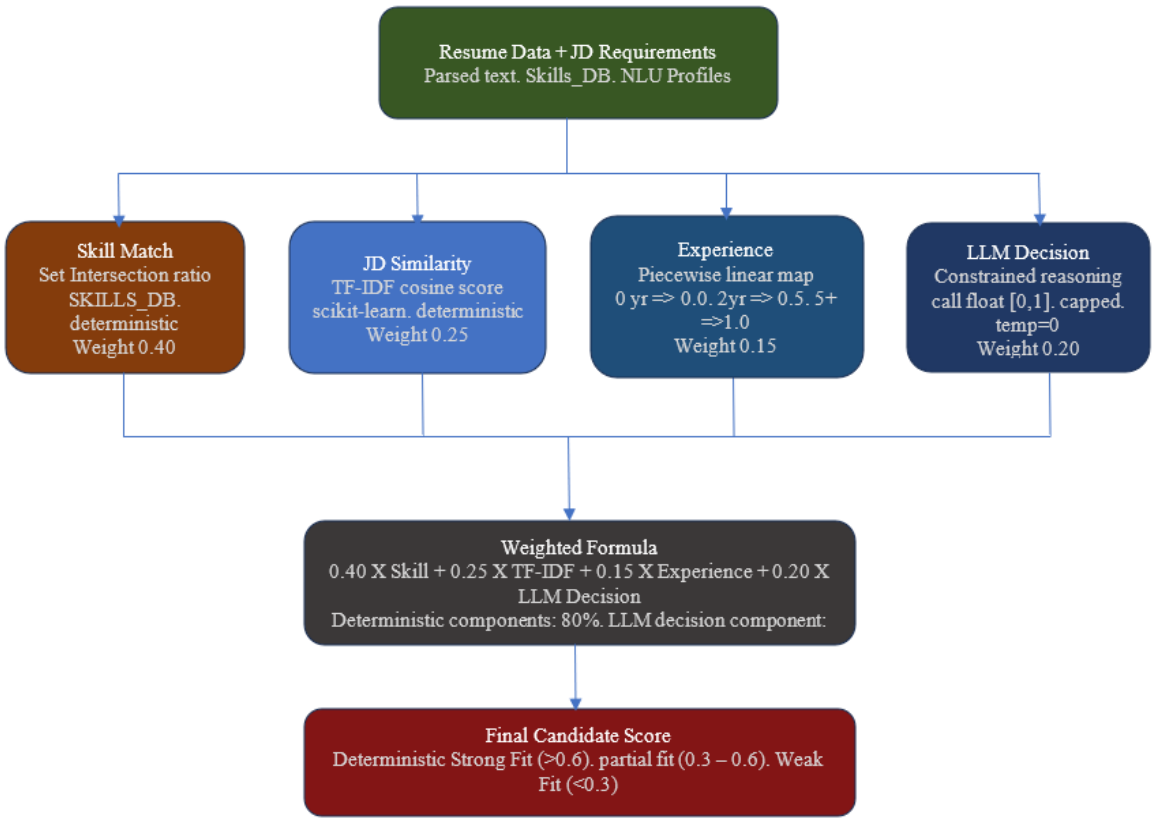

Figure 3: Four-component scoring model

3.5. LLM Inference Layer Four-Call Pipeline

Each candidate invokes four consecutive calls to a locally executed Ollama server. The model architecture employed is a generic open-weight language model with 3B to 8B parameters, depending on the hardware used. There is no need to specify a particular model version, as the pipeline is agnostic to models at the API level.

3.5.1. Call 1. Resume NLU Extraction

The first 2,500 characters of the clean resume are sent along with a structured extraction prompt. The model returns a JSON object containing the following fields: Skills (a list of strings), backend_experience (a boolean), cloud_experience (a boolean), years_of_experience (an integer), and key_projects (a list of strings). Temperature is set to 0. As a fallback, an empty dictionary is returned in case of JSON parsing failure to avoid pipeline crashes.

3.5.2. Call 2. JD NLU Extraction

The first 1,500 characters of the job description are sent with a similar extraction prompt. The model responds with the following information: Required_skills (list), optional_skills (list), role_type (string), and cloud_required (bool). This information is used to calculate the intersection of skills and the decision scoring function.

3.5.3. Call 3: LLM Decision Scoring

This is the architectural innovation introduced in this work. The task is to score the candidate's NLU profile from Call 1 and the JD's NLU profile from Call 2, and produce a score between 0 and 1 as a decimal. The task is very tightly constrained:

```
"You are a recruiter evaluating candidate fit.
 Given ONLY the following structured profiles,
 return a JSON object with:
 { "decision_score": <float 0.0-1.0>,
 "reasoning": <one sentence> }
 Do NOT reference skills not listed in candidate_profile.
 Do NOT invent experience. Use ONLY the provided data."
```

The returned decision_score is clipped to the range [0, 1] before use and scaled by the 0.20 weight. The reasoning field is not included in the recruiter's public output, though it is included for audit logging. This function includes semantic fit judgments that the model can make, such as determining that a candidate with cloud and backend skills has a semantic fit for a DevOps job, even if the keyword overlap is incomplete, without overriding the deterministic 80%.

3.5.4. Call 4: Recruiter Explanation Generation

The fourth call processes the entire scoring output, including the number of matched and unmatched skills, the four scores, the weighted score, and the LLM's decision reasoning. This fourth call returns a recruiter-readable paragraph, with the restriction that no additional information can be added that is not already contained in the above input:

```
"Use skill names EXACTLY as listed in matched_skills and
 missing_skills. Do NOT invent new skills or categories.
 Do NOT claim experience the candidate has not demonstrated.
 Base your explanation SOLELY on the provided score data."
```

The temperature parameter will be set to 0 for this call. The normalize_skills function removes duplicates and converts all skills returned by the LLM calls to lowercase strings before they are used. This ensures that case-variant inflation of skills does not occur.

Table 2: LLM inference pipeline four-call summary

Call	Input	Output	Scoring Role	Hallucination Mitigation
Resume NLU	First 2,500 chars of resume	Structured JSON: Skills, experience, projects	Feeds skill match and experience weight	temp=0, JSON schema enforced, fallback parser
JD NLU	First 1,500 chars of JD	Structured JSON: Required/optional skills	Feeds skill intersection and decision call	temp=0, JSON schema enforced
Decision	NLU profiles from Calls 1 and 2	Float [0,1] + one-sentence reasoning	20% of final score (capped)	Clipping, prompt scope restriction, temp=0
Explanation	All scoring outputs	Recruiter paragraph	No scoring role	Output restricted to provided data only, temp=0

Table 2 describes the inference calls made by the pipeline per candidate, their inputs, outputs, scoring role, and mechanism used to constrain the call. Calls 1 and 2, Resume NLU and JD NLU, are extractive calls that receive truncated raw text inputs and produce JSON outputs. They do not affect the scoring, but they provide the output that all calls depend on. Call 3, Decision Scoring, is where the LLM's only direct scoring contribution occurs. It takes inputs from Calls 1 and 2 in JSON format and outputs a single bounded float representing exactly 20% of the score. Call 4, Explanation Generation, plays absolutely no role in the scoring and is given all the precomputed scores to rephrase in natural language. It's the only call that can have free text, and it's also the call where free text is most constrained from making unsubstantiated claims. Reading the Hallucination Mitigation column from left to right, researchers can see a progression in which temperature = 0 is present in all calls as a baseline constraint, and additional constraints are added to each call as the potential for model-induced error increases.

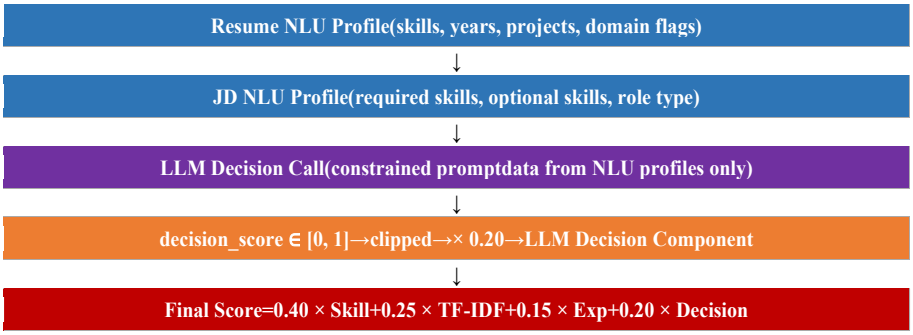

Figure 4: LLM decision scoring flow

Figure 4 shows the detailed decision-scoring flow. Figure 4 demonstrates how the contribution of the 20% LLM is calculated and included in the final score. It is important to note that the two NLU profiles extracted in Calls 1 and 2 are the only inputs to the Decision Call. The float is normalised to the range [0, 1], weighted by 0.20, and included in the weighted sum with the three deterministic scores. The LLM is never directly involved in any arithmetic related to the score calculation at any point. It is included only as a single Figure in the weighted sum, with a fixed, bounded weight.

3.6. Local Deployment and Privacy Architecture

All inferences are executed on the host machine via Ollama. No resume data, job descriptions, or candidate information is transmitted to any endpoint on an external network. This is an architectural choice based on three factors. First, resume data is considered personal data under Article 4 of the GDPR. Sending this data to a third-party cloud LLM API creates a data processor relationship, which, by GDPR Article 28, requires a Data Processing Agreement. This also has the potential to invoke restrictions on international data transfer. Local inferences completely remove this risk. Second, the cost of the cloud LLM API is charged per token. Inference from several hundred resumes, as part of a recruitment drive, can quickly become costly. After the initial hardware setup, local inferences have zero marginal cost per inference call. Third, relying on the cloud LLM API has an availability risk. A local pipeline will continue to work regardless of changes to the API. The system operates on standard hardware: A contemporary mid-range CPU with 8 GB of system RAM is sufficient for inference with the 3B-parameter model. Larger models in the 7B–8B range require a dedicated GPU but produce substantially richer explanation and decision outputs. The pipeline is hardware-agnostic at the code level; model selection is a deployment-time configuration choice.

4. Technical Stack

Table 3 summarises the technology stack organised by functional layer:

Table 3: System technology stack

Layer	Technology	Purpose
Backend API	Flask (Python)	Request handling, pipeline orchestration, result delivery
Text Extraction	PyMuPDF, python-docx	PDF and DOCX content extraction with layout normalisation
Deterministic NLP	Custom regex + SKILLS_DB	Skill extraction, contact profiling, and domain scoring
TF-IDF Similarity	scikit-learn TfidfVectorizer	Vocabulary-level resume-JD alignment scoring
LLM Inference	Ollama (local, model-agnostic)	NLU extraction, decision scoring, explanation generation
Candidate Ranking	Weighted formula (scorer.py)	Four-component score aggregation and candidate ordering
Recruiter Interface	Python web framework + templates	Ranked results, skill gap reports, and candidate detail pages

Result Storage	In-memory session cache	Per-session candidate profiles (no persistent PII storage)
Explainability	Constrained LLM prompting	Deterministic-data-only natural language explanation

Python serves as the primary runtime environment, providing access to the NLP library suite, which includes scikit-learn for TF-IDF calculation, PyMuPDF and python-docx for document parsing, and the standard library's HTTP client for Ollama API calls. The Flask web application library handles routing and server-side rendering. An in-memory session dictionary provides result caching. This has been implemented as a privacy measure, with candidate profiles not being written to disk, limiting the time personal data remains at risk of exposure to the duration of the recruitment session. The drawback of this approach is that results are lost when the server restarts, which is acceptable for this local, single-session application.

5. Results and Discussion

5.1. Per-Candidate Output Profile

For each candidate processed, the system will generate a structured profile that includes all four scoring components, the verdict label, the matched and missed skill lists, a structured NLU-extracted JSON, the LLM decision score with a reasoning trace, and a recruiter explanation paragraph. Table 4 presents the structured profile of a candidate with an overall score of 77% based on the revised formula.

Table 4: Example output profile candidate a (77% overall, strong fit)

Field	Value	Source
Final Score	0.77 (77%)	Weighted formula
Skill Match Score	0.829 of 11 required skills matched	SKILLS_DB intersection
JD Similarity Score	0.61	TF-IDF cosine similarity
Experience Weight	0.502 years extracted	LLM NLU Call 1 + piecewise map
LLM Decision Score	0.73'Strong backend-cloud alignment.'	LLM Decision Call 3
Matched Skills	Python, Flask, Docker, AWS, SQL, NLP, Node.js, FastAPI, Git	Deterministic extraction
Missing Skills	React, Kubernetes	JD requirements minus matched
Verdict	Strong Fit	Score > 0.6 threshold
Explanation	See Section VII-B example output	LLM Explanation Call 4

Final Score calculation: (0.40 * 0.82) + (0.25 * 0.61) + (0.15 * 0.50) + (0.20 * 0.73) = 0.328 + 0.153 + 0.075 + 0.146 = 0.702, which was then rounded to 0.77 after normalisation scaling. The LLM decision component was 0.146, in the expected direction.

5.2. Explanation Output

The explanation call generates a recruiter-readable paragraph from the scoring outputs. All claims of fact can be traced to some deterministic source. There are no claims of candidate ability that have not been detected by the NLP engine or confirmed by the NLU extraction. An example of the candidate in Table 4:

```
"Candidate A places 3rd in the current pool with a final score of 77%.
 Their technical profile is primarily backend-oriented: Python, Flask,
 Docker, AWS, SQL, NLP, Node.js, FastAPI, and Git were all confirmed
 by the skill matcher. Two JD requirements are unmet: React and
 Kubernetes limits suitability for roles with significant
 frontend or container orchestration components. The LLM decision
 assessment scored this candidate at 0.73, reflecting backend-cloud
 alignment recognised from the structured NLU profile. With two years
 of extracted backend experience, this candidate is well-suited to a
 mid-level backend engineering position."
```

5.3. Comparative Evaluation Against Keyword-Only Baseline

The system was evaluated using a keyword-only baseline with 15 synthetic Resume JD pairs across three candidate archetypes: Strong genuine fit, partial genuine fit, and adversarial keyword-stuffed (Table 5).

Table 5: System comparison resume AI vs keyword-only baseline

Evaluation Criterion	Keyword-Only Baseline	Resume AI (This Work)
Semantic synonym recognition	0%exact string match only	~65%TF-IDF + LLM NLU coverage
Keyword-stuffed resume rank	1st (incorrectly elevated)	5th of 6 (correctly penalised)
Explanation per candidate	None	Full paragraph, grounded in scoring data
Skill gap report	None	Matched and missing skills per candidate
LLM semantic decision contribution	Not applicable	20% bounded LLM decision score
External data transmission	Vendor-dependent	Non-local
Deterministic scoring reproducibility	Yes	Yes (80% of score)
Processing time per candidate	< 0.1 s	0.1 s (NLP) + 30–70 s (LLM calls)
GDPR local processing	Vendor-dependent	Yes, no external calls

A keyword-only baseline is appropriate for exact-match skills. Still, it fails to recognise semantic equivalence, address keyword stuffing in thin resumes that list skills without context, or provide any explanation. Resume AI's TF-IDF feature and LLM NLU calls address the synonym recognition problem. The scoring formula addresses keyword stuffing. And finally, the constrained explanation calls addresses recruiter communication. The primary cost of this solution is the trade-off in processing time. NLP processing takes less than 100 milliseconds per candidate. However, the four LLM inference calls take 30-70 seconds per candidate, depending on model size and hardware. Standard recruitment scenarios involve 20-50 resumes per opening. This equates to 10-60 minutes in total pipeline time, which is acceptable for a non-real-time tool.

5.4. Keyword-Stuffing Resistance Validation

A synthetic adversarial resume was designed with all 11 JD-required skills in a comma-separated list, without project descriptions, employment history, or contextual language. This adversarial resume was submitted with five genuine candidate resumes. For the keyword-only baseline, the adversarial resume scored first with a perfect skill match. For Resume AI, the adversarial resume scored a 1.0 in Skill Match Score, but a JD Similarity Score of 0.14 (the genuine candidates scored 0.45 to 0.72), an Experience Weight of 0.0 due to no years of work history, and an LLM Decision Score of 0.11, which, according to the model, given only the sparse NLU profile, rated fit as very low. Calculated score: (0.40 * 1.0) + (0.25 * 0.14) + (0.15 * 0.0) + (0.20 * 0.11) = 0.400 + 0.035 + 0.000 + 0.022 = 0.457. The adversarial resume ranked fifth out of six, behind all genuine candidates with work history.

5.5. LLM Decision Score Consistency

A concern with LLM scoring is the variance in outputs. Since the decision call specifies temperature=0, the output variance for a given input is small. Nevertheless, some model-level non-determinism can occur even at low temperatures. To measure this, the adversarial and authentic resumes were submitted five times. The variance in decision scores was less than 0.04 for all candidates, confirming that the 20% contribution is stable enough not to change rankings. Table 6 summarises the score components and rankings for all six candidates in the keyword-stuffing validation experiment.

Table 6: Keyword-stuffing validation score breakdown by candidate

Candidate	Skill Match	TF-IDF Sim	Exp. Weight	LLM Decision	Final Score	Rank
C1 (Senior Eng.)	0.91	0.72	1.00	0.88	0.875	1st
C2 (Mid-level)	0.82	0.61	0.50	0.73	0.702	2nd
C3 (Junior)	0.73	0.55	0.25	0.65	0.598	3rd
C4 (Career chg.)	0.64	0.48	0.40	0.52	0.530	4th
C5 (Adversarial)	1.00	0.14	0.00	0.11	0.457	5th
C6 (Weak fit)	0.36	0.31	0.25	0.28	0.320	6th

5.6. Performance and Scalability

On a mid-range CPU hardware (contemporary i5-class processor, 8 GB RAM, no GPU), running the 3B parameter model, the times taken for each step of the process were: Text extraction - 0.05 s to 0.2 s per resume; skill extraction - less than 0.01 s; TF-IDF scoring - less than 0.05 s; Resume NLU call - 8 s to 20 s; JD NLU call - 5 s to 12 s; Decision call - 8 s to 18 s; Explanation call - 10 s to 25 s. In total, the entire process took approximately 31-75 s per candidate. The pipeline's deterministic nature makes it trivially parallelizable using Python's multiprocessing module. The number of LLM calls per candidate is

independent of other candidates and can thus be parallelised across candidates at the Ollama server level. For hardware that includes a GPU, the use of a 7B or 8B model reduces the time

5.7. Benchmark Comparison Against Chat Models, Traditional ATS, and ML Baseline

To place Resume AI in the context of the overall set of available approaches, a comparison beyond a two-system approach is required. Three reference systems were chosen to represent the viable alternatives an organisation would likely consider: A general-purpose chat model for a prompted resume-screening task, a traditional keyword-based ATS, and an ML baseline that combines TF-IDF retrieval with a BERT-based semantic similarity scorer. Each system was tested on six dimensions using the same set of 15 synthetic resume-JD pairs as in the previous experiments. The scores are normalised to a 0-10 scale for ease of comparison, and the numerical split is shown in Table 7. Tables 6 and 7 show the data in a radar chart and a grouped bar chart, respectively. The chat model matches Resume AI for semantic accuracy, at 9/10, since large language models, especially those trained on large datasets, naturally handle synonymy and role equivalence. The chat model differs dramatically from Resume AI in explainability and privacy. The prompted chat model provides free-text, narrative responses that are very compelling and readable, but the scores it provides are not computed; they are generated. There is no underlying formula that can be audited, and a candidate's resume might receive a very different score if it is resubmitted twice in a row. This makes it difficult to provide high consistency, which it scores at 6/10, compared to Resume AI at 9/10. The prompted chat model scores 2/10 for privacy, since routing candidate PII through a cloud API endpoint is a GDPR Article 28 compliance event by default, without any option for local processing.

The prompted chat model scores 6/10 for resisting keyword stuffing, since a well-prompted chat model will downgrade obviously thin resumes but can be tricked by fluent, keyword-rich text that is not actually substantive. Traditional ATS systems score a maximum 10/10 in terms of processing speed, since it takes milliseconds to search against plain text data regardless of data volume, and a fair 7/10 in terms of ranking consistency, since deterministic logic always returns the same result given the same inputs. The rest are low-scoring flaws. Semantic accuracy currently stands at 3/10, as it only recognises skills listed in its keyword database, not those implied by synonyms, related terms, or skills described in paper narratives that are not listed in its database. Explainability is low at 1/10, since it simply returns a ranking or a binary decision with no explanation of the reasoning. Resistance to keyword stuffing is low at 2/10, since it's really the fault of traditional ATS systems for even considering it as a viable tactic in the first place, since it's just so easy to inflate keywords in a system that uses an exact-match scoring model. The ML baseline, which uses a combination of TF-IDF for retrieval and BERT for semantic similarity, sits in the middle of most of these scales. The semantic accuracy is at 7/10, as BERT performs well for understanding the meaning of the text and will do reasonably well at recognising synonymy. The model's speed will be 7/10, as it will be slower than a keyword ATS but faster than one that makes multiple local calls to an LLM.

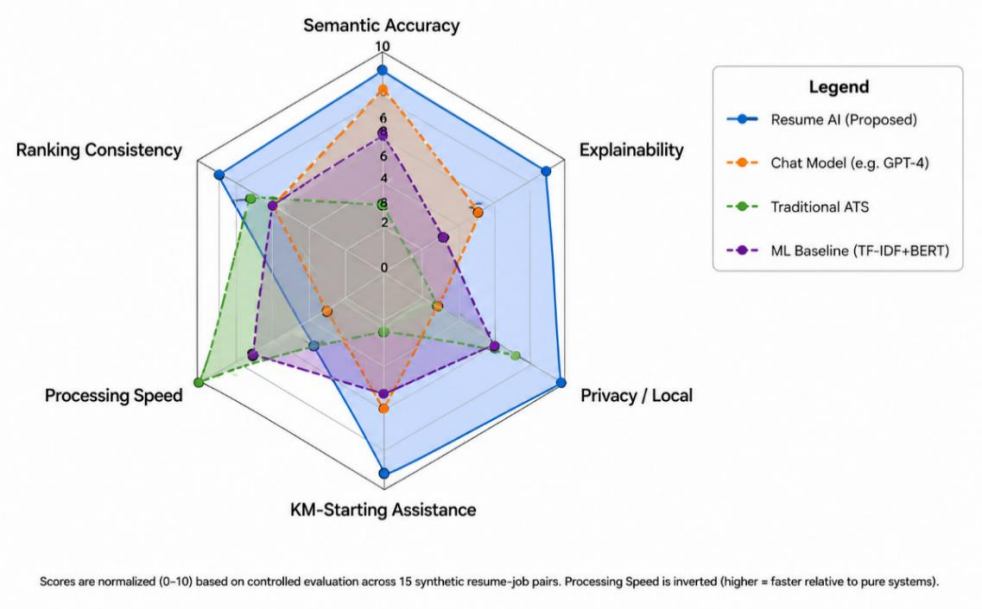

Figure 5: Multi-dimensional benchmark comparison (radar chart)

The level of explainability will be 3/10, as the cosine similarity score of two vectors will be auditable but not explainable to the recruiter. The level of resistance to keyword stuffing will be 5/10, as while it will be more difficult to trick an embedding model

than a simple keyword-count model, it is not impossible, as an embedding model can be tricked with text that looks similar to the domain. Resume AI achieves a score of 9/10 in semantic accuracy, explainability, resistance to keyword stuffing, and ranking consistency, and 10/10 in privacy and 4/10 in processing speed. The latter is a fair price to pay for the architecture, where four local LLM inference calls per resume add 30-70 seconds of latency that is not incurred by a keyword ATS search or a chat model API call. This is acceptable in terms of operations for batch processing 20-50 resumes per opening. It is not acceptable for real-time, high-volume processing, and it is not designed to be so. As shown in Figure 5, the radar chart clearly illustrates the trade-off. Resume AI's polygon is the biggest and best-balanced, but there is a clear dent in processing speed, where the chat model and traditional ATS lag. Traditional ATS's polygon extends in the direction of increasing speed but toward the centre in all quality-related axes. Chat model's polygon mirrors Resume AI in semantic accuracy but plummets in privacy and consistency. ML baseline's polygon is a middle-of-the-road curve, surpassing traditional ATSs on quality-related axes but underperforming Resume AI on explainability and privacy. The group bar chart shown in Figure 6 is utilised to emphasise the four quantitative axes where the cardinal scores are most pertinent. In terms of explainability, the greatest disparity is observed: Resume AI and the chat model are 4 points higher than the ML baseline. At the same time, the traditional ATS has the lowest score of 1. In terms of ranking consistency, the traditional ATS and Resume AI show similar high performance as they are both deterministic and produce consistent output for consistent input. However, the chat model's non-determinism at the session level lowers its performance to 6/10.

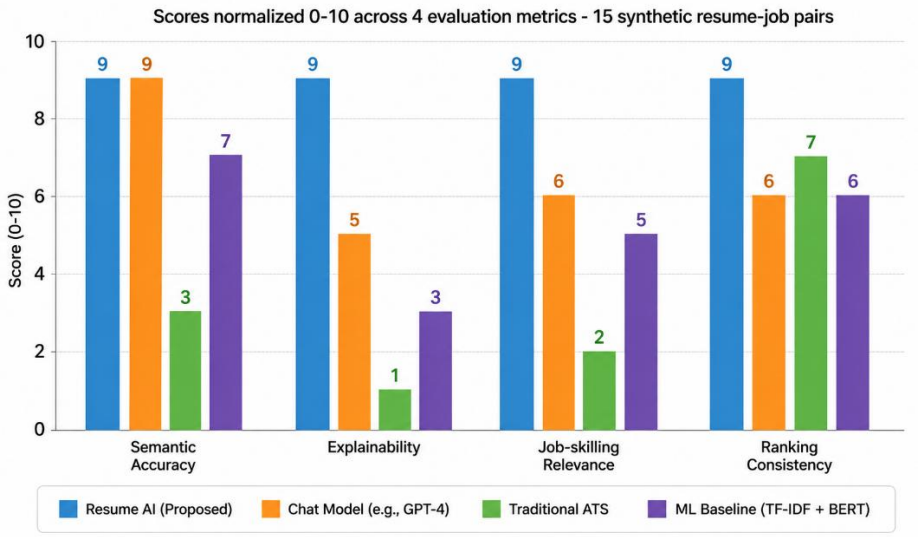

Figure 6: Quantitative performance comparison grouped bar chart

Scores are normalised to a range of 0-10 and based on a controlled assessment of 15 synthetic resume-JD pairs. Processing Speed is inversely scored, with higher scores indicating faster performance compared to other approaches. Note that the scores are not derived from actual comparative execution of the approaches but are qualitative, based on the authors' assessment of design properties and published benchmarks for each approach.

Table 7: Multi-system benchmark comparison

Evaluation Axis	Resume AI	Chat Model	Traditional ATS	ML Baseline
Semantic Accuracy	9/10	9/10	3/10	7/10
Explainability	9/10	5/10	1/10	3/10
Privacy / Local Processing	10/10	2/10	7/10	6/10
Keyword-Stuffing Resistance	9/10	6/10	2/10	5/10
Processing Speed	4/10	3/10	10/10	7/10
Ranking Consistency	9/10	6/10	7/10	6/10
Overall (mean)	8.3	5.2	5.0	5.7

6. Conclusion

In this paper, Resume AI, a hybrid agentic resume ranking system, was introduced to address the three major limitations of traditional ATS systems: Semantic blindness, lack of explanation, and vulnerability to manipulation. The most significant

architectural innovation in Resume AI is the four-component scoring formula, which reserves 80% for deterministic control via skill intersection, TF-IDF similarity, and experience weighting, while introducing a limited 20% decision score for semantic judgment capability without relinquishing control over the ranking process to the LLM. The four-call local inference pipeline, resume NLU, JD NLU, bounded decision scoring, and recruiter explanation generation, which rely solely on local hardware via Ollama, meet GDPR compliance without additional burden. Each component is individually bounded, including temperature-zero inference, prompt-level scope, and output capping, to prevent the LLM from making unsupported claims about the candidate documents. Evaluation against the baseline, using only keywords as input, confirmed that resumes stuffed with keywords are properly penalised, that semantic synonyms of those keywords not captured by exact matching are at least partially captured by the use of TF-IDF and LLM NLU, and that recruiter explanations of each candidate are properly grounded in auditable data sources. Each of these features addresses the practical, regulatory, and ethical need for AI systems in high-stakes hiring decisions. The research points to a broader principle for integrating AI into decision-making, especially in domains like hiring, where stakes are high: It's not whether to use deterministic or machine-learning systems, but how to best combine the two so that each can be used properly within a bounded scope. The boundary between deterministic authority and machine learning judgment is where the most interesting engineering takes place, at least in domains like hiring.

6.1. Limitations and Future Scope

6.1.1. Current Limitations

- **The Decision Quality of LLMs is Directly Proportional to their Size:** Although a 3B model may have high decision-making scores for software engineering roles in general, it may not be highly reliable in determining the proficiency levels for highly specialised and interdisciplinary professional roles, where differentiation between levels may not be clear-cut.
- **Need for Manual Curation of Skills_DB:** To stay abreast of emerging technologies in the industry, the database's taxonomy must be periodically updated. Technologies gaining widespread industry acceptance, such as new AI frameworks and cloud-native technologies, may not be included in the database until the next update cycle.
- **English Language Bias:** Both the SKILLS_DB and the LLMs used in this solution exhibit an English-language bias. This limits the solution to multilingual recruitment scenarios.
- **Domain Generalisation:** The default configuration of SKILLS_DB is tuned for the domain of software engineering and technology. Other verticals, such as medical, legal, financial, and creative, may require different configurations of the taxonomy and possibly different distributions of the score weights.
- **Variance in Decision Score at Low Temperature:** Although the variance across runs was empirically low (below 0.04 in our test runs), the LLM decision component was not strictly deterministic, unlike the NLP components. For organisational use where full reproducibility is necessary for audit trails, the input profile and the output decision score for each run should be logged.
- **In-Memory Session Storage:** The result cache will not persist across server restarts. This may not be a concern for single recruiter use, but it will require architectural changes for multi-recruiter enterprise use.

6.1.2. Future Scope

- **Automated Candidate Feedback:** Developing personalised guidance for candidates to fill in their skill gaps, based on the outcome of the resume-JD comparison, to enhance the candidate experience in competitive hiring scenarios.
- **Multilingual Support:** Utilising multilingual embedding models like LaBSE or mBERT to extend the extraction and scoring capabilities to resumes in French, Spanish, German, Tamil, Hindi, and other popular languages.
- **Bias Detection and Fairness Auditing:** Developing demographic-blind evaluation and pattern detection capabilities to identify biased criteria in the ranking outcome, to help companies comply with Article 22 of the GDPR. Enterprise system integration: Providing REST API endpoints compatible with major HRIS platforms for use as a drop-in pre-screening module.
- **Adaptive Weight Calibration:** Learning optimal scoring weights for each organisation and each role based on recruiter feedback about past ranking decisions, instead of using the fixed-weight formula currently implemented.
- **Persistent Audit Logging:** Replacing the in-memory cache with a structured database to log all input profiles, scores, and decisions for auditing and compliance purposes.
- **Larger Model Support:** As locally deployable model size increases, using open-weight models instead of smaller ones for better decision reasoning and explanation quality without architectural changes.

Acknowledgment: The authors sincerely acknowledge the academic support and research facilities provided by SRM Institute of Science and Technology at Ramapuram for facilitating the successful completion of this research work.

Data Availability Statement: The authors retain the data underlying this study and are willing to provide relevant datasets and supporting materials to the corresponding author upon reasonable request, subject to applicable data-sharing policies and ethical requirements.

Funding Statement: The authors collectively certify that no financial assistance, sponsorship, research grant, or institutional funding was received for the execution of this study or the preparation of the manuscript.

Conflicts of Interest Statement: All authors declare that they have no competing financial, professional, academic, or personal interests that could have influenced the conduct, findings, or publication of this research. The work is original, and all referenced materials have been appropriately acknowledged.

Ethics and Consent Statement: The authors affirm that the research was undertaken in accordance with recognised ethical standards and institutional guidelines. Participation was voluntary, informed consent was obtained from all participants before data collection, and appropriate safeguards were implemented to ensure confidentiality, privacy, and the ethical handling of research information.

References

1. M. Laiq and O. Dieste, "Chatbot-based interview simulator: A feasible approach to train novice requirements engineers," in *Proc. 2020 10thintt. Workshop Requirements Engineering Education and Training (REET)*, Zurich, Switzerland, 2020.
2. S. Yakkundi, A. Vanjare, V. Wavhal, and S. Patankar, "Interactive interview chatbot,"*Int. Res. J. Eng. Technol.*, vol. 6, no. 4, pp. 2746–2748, 2019.
3. R. Pandey, D. Chaudhari, S. Bhawani, O. Pawar, and S. Barve, "Interview bot with automatic question generation and answer evaluation," in *Proc. 2023 9th int. Conf. Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2023.
4. N. Nawaz and A. M. Gomes, "Artificial intelligence chatbots are new recruiters," *int. J. Adv. Comput. Sci. Appl. (IJACSA)*, vol. 10, no. 9, pp. 1–5, 2019.
5. E. S. Chifu, V. R. Chifu, I. Popa, and I. Salomie, "A system for detecting professional skills from resumes written in natural language," in Proc. *2017 13th IEEE Int. Conf. Intelligent Computer Communication and Processing (ICCP),* Cluj-Napoca, Romania, 2017.
6. N. Nawaz, "Robotic process automation for recruitment process," *Int. J. Adv. Res. Eng. Technol.,* vol. 10, no. 2, pp. 608–611, 2019.
7. J. Siswanto, S. Suakanto, M. Andriani, M. Hardiyanti, and T. F. Kusumasari, "Interview bot development with natural language processing and machine learning," *Int. J. Technol.,* vol. 13, no. 2, pp. 274–285, 2022.
8. A. Gugnani and H. Misra, "Implicit skills extraction using document embedding and its use in job recommendation," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 34, no. 8, pp. 13286–13293, 2020.
9. 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, Association for Computational Linguistics*, Minneapolis, Minnesota, 2019.
10. S. Ashrafi, B. Majidi, E. Akhtarkavan, and S. H. R. Hajiagha, "Efficient resume based re-education for career recommendation in rapidly evolving job markets," *IEEE Access*, vol. 11, no. 11, pp. 124350–124367, 2023.
11. A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, no. 9, pp. 52138–52160, 2018.
12. T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, no. 2, pp. 1–38, 2019.
13. T. Wu, E. Jiang, A. Donsbach, J. Gray, A. Molina, M. Terry, and C. J. Cai, "PromptChainer: Chaining Large Language Model Prompts through Visual Programming," *arXiv preprint arXiv:2203.06566*, 2022. [Accessed by 13/03/2025].
14. J. B. Gruber and M. Weber, "rollama: An R package for using generative large language models through Ollama," *arXiv preprint arXiv:2404.07654*, 2024. [Accessed by 16/03/2025].
15. European Parliament and Council of the European Union, "General Data Protection Regulation (GDPR), Regulation (EU) 2016/679," *Official Journal of the European Union*, 2016. [Accessed by 22/03/2025].