Single-Token Expected-Value Scoring for Cold-Start Candidate Ranking

Qihang Wang[1], Jinwei Tan[1], Mengyuan Shi[1], Mayank Sharma[1], Shuai Zhao[1], Fuxian Li[1], Ryan Yan[1], Alexander P. Kreuzer[1], Mohit Jain[1], Dheeraj Toshniwal[1] and Manoj Seethamsetty[1]

[1]*Indeed, Inc.*

Abstract

AI-assisted sourcing helps streamline the initial stages of candidate review, reducing the administrative burden of manual screening for recruiters. However, deploying language models as production rankers remains challenging. Zero-shot Large Language Models (LLMs) may produce unstable, non-deterministic scores and provide limited ranking accuracy, while conventional deep neural rankers require large interaction logs that are often unavailable in cold-start settings. We use *cold start* in a specific sense. Fitting a deep neural ranker for recruitment from scratch requires orders of magnitude more logged interaction data than a low-traffic, specialized sourcing platform produces; in our experience training such rankers in production, on the order of millions of interactions. That behavioral signal is what our setting lacks. What is available instead is a few hundred thousand ordinal relevance labels, which are small by ranker-training standards but sufficient here because a pretrained language model already encodes the general world knowledge the task depends on, allowing fine-tuning to supply domain alignment. We present *single-token expected-value scoring*, a ranking primitive that casts candidate–job relevance as an ordinal classification over the grade tokens $\{1, \ldots, 5\}$ and reads the relevance score as the mathematical expectation of the first-token probability distribution. Because the score is derived from a single decoding step rather than open-ended generation, it is a deterministic function of the model's output logits, requires no output parsing, and serves at low latency. To let this primitive learn the non-linear interdependencies of heterogeneous hiring criteria from this supervision alone, we fine-tune a Small Language Model (SLM) with a hybrid ordinal regression loss that combines a Mean Squared Error (MSE) term, preserving ordinal distance with a categorical Cross-Entropy (CE) term that sharpens class boundaries. The resulting framework is data-efficient, converging in cold-start, low-traffic regimes without the large interaction logs deep neural rankers demand. We assess ranking quality along two relevance dimensions — Jobseeker Relevance (how well a job matches a jobseeker's preference) and Employer Relevance (how well a jobseeker meets an employer's requirements) — using NDCG@10 and low relevance rate (the fraction of top results that are poor matches). Offline, our fine-tuned model outperforms the heuristic baseline and zero-shot LLMs. An end-to-end simulation shows the same direction at larger magnitude, with a 54.2% increase in Jobseeker Relevance NDCG@10, a 46.7% reduction in low relevance rate, and consistent Employer Relevance gains; these simulation figures are large partly because the heuristic baseline optimizes only the employer side, leaving substantial jobseeker-side headroom. In a live online experiment it reduces employer low-relevance by 27.3% and raises the employer keep rate by 7.07%. Our work offers a stable and efficient ranking method that remains competitive under severe data constraints.

Keywords

Single-Token Scoring, Expected-Value Ranking, Cold Start Candidate Ranking, Ordinal Regression, Small Language Models, LLM4Rec

1. Introduction

Hiring teams often review many resumes and profiles to decide which candidates to contact, a process known as candidate sourcing. AI systems are increasingly used to improve this process and reduce manual screening [1]. Because both candidate profiles and job descriptions are composed of rich, unstructured natural language, the domain is well-suited to Large Language Models (LLMs) and Small Language Models (SLMs). A growing number of commercial products and academic frameworks now leverage these generative models to reason about the semantic alignment between candidates and open roles [2, 3].

However, training ranking models in this domain presents empirical challenges. Traditional neural network-based approaches typically require vast amounts of historical interaction data to adequately fit the complex distributions of recruitment data [4]. Emerging platforms or low-traffic products typically lack historical interaction data, making it difficult to train machine learning models, and handcrafted heuristic methods are used as an alternative [5]. We use *cold start* in this specific sense throughout: the constraint is the volume of behavioral signal, not the availability of supervision. Training a deep neural ranker from scratch on recruitment data requires interaction logs orders of magnitude larger than a specialized, low-traffic platform accumulates, whereas ordinal relevance labels can be generated on demand by an LLM labeler (Section 3.2). A few hundred thousand such labels remain small by ranker-training standards, and fine-tuning succeeds at that scale only because the pretrained backbone already supplies the general world knowledge the task depends on. Our results bear this out: for the GPT-4.1-mini variant, gains from 20k to 50k labels are already small (Section 4), and the binding constraint proves to be the capacity of the base model rather than the volume of task data (Section 7).

By delegating semantic rule evaluation and multi-dimensional criteria judgment to LLMs, hybrid frameworks that pair these models with traditional heuristics have become common [6]. Within these pipelines, LLMs parse unstructured text to verify specific job requirements, connecting rigid algorithmic rules with flexible semantic understanding.

Despite the popularity of these LLM-heuristic hybrid approaches, two bottlenecks persist. First, the individual criteria evaluated by the LLM and the heuristic components rarely aggregate in a simple linear fashion, making it difficult to capture the nuanced, non-linear interactions required for optimal candidate ranking. Second, when relying on zero-shot LLMs, the models are frequently relegated to binary hard-filtering of explicit constraints rather than executing

RecSys in HR '26: The 6th Workshop on Recommender Systems for Human Resources, in conjunction with the 20th ACM Conference on Recommender Systems, September 28–October 2, 2026, Minneapolis, MN, United States.
wqihang@indeed.com (Q. Wang); jint@indeed.com (J. Tan); mshi@indeed.com (M. Shi); masharma@indeed.com (M. Sharma); szhao@indeed.com (S. Zhao); fuxianl@indeed.com (F. Li); myan@indeed.com (R. Yan); akreuzer@indeed.com (A. P. Kreuzer); mjain@indeed.com (M. Jain); dtoshniwal@indeed.com (D. Toshniwal); manojs@indeed.com (M. Seethamsetty)

continuous, calibrated ranking [7]. These zero-shot LLMs, accessed via APIs, also introduce consistency issues; the stochastic nature of LLMs often yields volatile results across different API calls, a vulnerability that becomes particularly pronounced when processing large token inputs or generating open-ended responses [8].

In this work, we propose *single-token expected-value scoring* for cold-start candidate ranking. Our contributions are:

- **A deterministic ranking primitive.** We score candidate–job relevance as the expected value of the first-token probability distribution over the ordinal grade tokens $\{1, \ldots, 5\}$. Computing the score from the first-token logits in a single decoding step makes it deterministic, removes output-parsing failures, and keeps inference latency low — directly addressing the volatility of open-ended LLM generation.
- **A data-efficient training recipe for cold start.** We fine-tune an SLM with a hybrid ordinal regression loss (MSE + CE) that enables the single-token score to capture how heterogeneous hiring criteria interact non-linearly. The recipe converges on LLM-generated relevance labels alone, without the large interaction logs that deep neural rankers require.
- **Evidence from offline and online experiments.** Across offline ranking metrics, an end-to-end sourcing simulation, and a live online experiment, the method outperforms both the heuristic baseline and zero-shot LLMs, and we analyze where and why the gains arise (dual-sided relevance, ordinal integrity, and data scaling).

We validate the efficacy of our method by comparing it against both zero-shot LLM baselines and established heuristic scoring systems. Experimental results demonstrate that our proposed framework not only yields better alignment and predictive accuracy in low-data regimes but also maintains the computational stability required for real-world deployment, where it has been successfully integrated into a production talent sourcing platform.

2. Background & Problem Setting

We first describe the automated sourcing pipeline and formalize the ranking problem. The production recruitment system is structured as a decoupled, three-stage pipeline designed to help recruiters navigate large candidate pools efficiently. The pipeline begins with a retrieval phase, where an LLM performs strategic planning based on the recruiter's explicit requirements to formulate structured queries [9], consolidating the outputs into an initial candidate pool at the scale of thousands of candidates. Because this initial stage leverages the platform's legacy infrastructure, its internal mechanisms remain outside the scope of this paper.

Following retrieval, the core candidate evaluation occurs during the criteria judgment and scoring phase, which serves as the primary focus of this work. In this phase, an LLM parses the employer's unstructured job specification into distinct required and preferred criteria. The system then evaluates the candidate profile against each individual criterion and aggregates these granular judgments into a single relevance score using a handcrafted heuristic. This specific method of decomposing complex, unstructured documents into explicit, multi-dimensional attributes for individual evaluation has been validated in recent literature as an effective strategy to improve alignment precision and interpretability of text-matching systems [10]. In the final stage, the top-N candidates aggregated from multiple retrieval streams undergo global listwise reranking, accompanied by the generation of natural language insights, a module that is excluded from our present optimization scope.

The primary technical bottleneck of this architecture lies in the second stage. The heuristic implicitly treats overall candidate relevance as a linear, separable function of independent criteria, operating under the assumption that individual scores can simply be linearly added. In real-world talent acquisition, however, true relevance is inherently non-linear and characterized by complex interdependencies among qualifications, where the satisfaction of one critical requirement may heavily outweigh or alter the significance of others.

Formulating a robust solution to this problem presents a behavioral-signal constraint rather than a supervisory one. While training a standard deep neural ranker from scratch could theoretically capture these non-linear interactions, it is infeasible in this context. Sourcing criteria are highly dynamic, varying continuously across different job roles and user sessions, and the low-traffic nature of specialized recruitment platforms yields sparse interaction signals.

Relying on zero-shot LLMs to perform this aggregation introduces substantial variance. Without task-specific alignment, these models are inconsistent: they produce conflicting reasoning and different scores for identical candidate contexts, failing to meet the consistency constraints of the task. This instability is well documented. Repeated calls on unchanged inputs yield different outputs even at low temperature [8], and LLM scorers are systematically sensitive to presentation factors that carry no information about the item being judged, such as the position an item occupies in the prompt [11]. Graded relevance prompting reduces but does not remove this variance [7].

We therefore reformulate the task as an ordinal classification problem with a fine-tuned SLM. Given the parsed required and preferred criteria alongside the unstructured candidate profile, the SLM is optimized to map the input context directly to a discrete ordinal relevance label within the set $\{1, 2, 3, 4, 5\}$, which subsequently dictates the intra-job candidate ranking. We fine-tuned the model to align with complex marketplace preferences while keeping the scoring consistency and low latency that deployment requires.

3. Approach

3.1. Heuristic Baseline

During onboarding, each job's requirements are decomposed into a set of required and preferred criteria, either provided by the employer or generated from the job description. For a given candidate, an LLM judges each criterion independently, assigning an ordinal fulfillment level (ranging from explicitly satisfied to not met) based on the evidence in the candidate profile. Each level is mapped to a fixed scalar, and the per-criterion scores are combined into a single relevance score via a weighted linear sum, with required and preferred criteria weighted differently. Candidates are then thresholded on this aggregate score to form the set passed to downstream listwise reranking. This design embodies the

additive separability assumption we challenge in this work: overall relevance is treated as a linear sum of independently scored criteria, with no mechanism to model interactions between them (Figure 1).

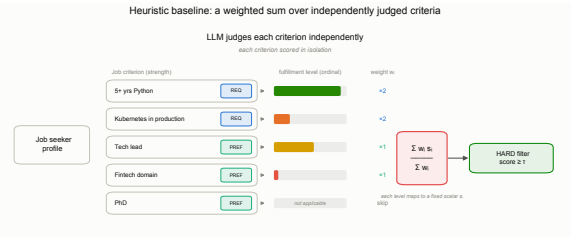

Figure 1: Heuristic baseline. A job's requirements are decomposed into required and preferred criteria; an LLM judges the candidate against each criterion independently, and the per-criterion scores are aggregated into a single relevance score by a weighted linear sum before thresholding.

3.2. Single-Token Expected-Value Scoring

We follow the labeling framework of Pei et al. [12]. For each job–jobseeker pair, an LLM labeler produces two complementary relevance signals: *Employer Relevance*, which measures how well the jobseeker satisfies the employer's requirements, and *Jobseeker Relevance*, which measures how well the job aligns with the jobseeker's preferences or likely interests. A fixed, deterministic policy maps the employer-side verdict and the jobseeker-side grade onto a single overall-relevance training label $y \in \{1, 2, 3, 4, 5\}$: 5 when both sides are the best fit for one another and 1 when either side is a bad match. Observed interaction outcomes take precedence where available. The policy is applied identically across the train, validation, and test splits, so no label drift is introduced across the time-based split of Section 4.

Here we use a pointwise approach, where the model produces a single score based on a pair of job and jobseeker. With the above labels, it is common to constrain the model to emit a single grade token and use the token probability as the ranking score. Our formulation is motivated by Zhuang et al. [7], who show that prompting an LLM ranker over a fine-grained ordinal label set and reading the *expected value* of the resulting distribution is substantially more effective than the conventional binary yes/no relevance probability, as the graded scale exposes finer distinctions that a single relevant-token probability cannot. Building on this idea, instead of taking the most likely grade we compute the *Expected Relevance* (ER): the expectation of the ordinal grade under the model's predicted distribution over the grade tokens. We formalize this expected-value score in Eq. (2) and use it both as the inference-time ranking score and as the regression target of our training loss.

Our departure from Zhuang et al. [7] is that they apply the expected-value score in a purely zero-shot prompting setting, whereas we turn it into a learnable training objective: we fine-tune the model with a hybrid ordinal regression loss (detailed below) that directly optimizes this expected-value score, combining a regression term with a classification term rather than relying on the pretrained distribution alone.

We experimented with both LLMs and SLMs:

- For the LLM, we take the first-token distribution, keep only the probabilities of the grade tokens 1–5 (renormalized), and compute their expected value. Standard commercial APIs support `logprobs` and `logit_bias`, allowing us to constrain the output to a single grade token and read its probability distribution.
- For the SLM, we do not need to specify the output token via `logit_bias`; we still calculate the final value using the expected-value method.

To effectively capture the inherent numerical and ordinal relationships in score-based token classification tasks (e.g., predicting ratings from 1 to 5), we utilize a hybrid *Ordinal Regression Loss*. The objective function consists of a joint optimization framework combining a Mean Squared Error (MSE) regression component with a standard Cross-Entropy (CE) classification penalty.

Let N denote the number of valid target samples in a batch, and let K represent the total number of ordinal classes. For each sample $i \in \{1, 2, \ldots, N\}$:

- $\mathbf{z}_i = [z_{i1}, z_{i2}, \ldots, z_{iK}]^\top \in \mathbb{R}^K$ represents the logit vector extracted from the sequence position immediately preceding the target token.
- $\mathbf{v} = [v_1, v_2, \ldots, v_K]^\top \in \mathbb{R}^K$ denotes the static vector mapping continuous score values to each class (e.g., $\mathbf{v} = [1.0, 2.0, \ldots, 5.0]^\top$).
- $y_i \in \{1, 2, \ldots, K\}$ is the ground-truth class index.
- $s_i = v_{y_i}$ represents the true scalar score value corresponding to the ground-truth class.

The categorical probability p_{ik} that the i-th sample belongs to class k is derived via the softmax operation over the target logits:

$$p_{ik} = \frac{\exp(z_{ik})}{\sum_{j=1}^{K} \exp(z_{ij})}. \tag{1}$$

Using the predicted class probabilities as weights, we compute the continuous *expected score* $\hat{s}_i$ for sample i as

$$\hat{s}_i = \sum_{k=1}^{K} p_{ik} \cdot v_k. \tag{2}$$

This expected score is exactly the Expected Relevance (ER) introduced above: it serves as the regression target in the loss below and, at inference time, as the ranking score. With the linear value map $v_k = k$, it reduces to $\hat{s}_i = \sum_{k=1}^{K} k\, p_{ik}$.

Mean Squared Error (MSE) Loss. The regression component minimizes the squared Euclidean distance between the continuous expected score and the true continuous score. This explicitly forces the network to respect the ordinal distance between distinct rating levels:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (\hat{s}_i - s_i)^2. \tag{3}$$

Cross-Entropy (CE) Loss. To maintain stable class discrimination and accelerate alignment, a standard categorical cross-entropy loss is simultaneously optimized:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \log p_{i,y_i}. \tag{4}$$

The final joint loss function $\mathcal{L}_{\text{total}}$ is formalized as a weighted sum of the two complementary objectives:

$$\mathcal{L}_{\text{total}} = w_{\text{MSE}} \cdot \mathcal{L}_{\text{MSE}} + w_{\text{CE}} \cdot \mathcal{L}_{\text{CE}}, \tag{5}$$

where w_{MSE} and w_{CE} are hyper-parameters controlling the regularization trade-off between the expected-value distance and strict categorical classification performance. Our deployed Qwen3-8B SFT model uses $w_{\text{MSE}} = 0.6$ and $w_{\text{CE}} = 0.4$; we ablate this choice against the two single-objective extremes in Section 4.7.

3.3. Fine-Tuning Configuration

We fine-tune the Qwen3-8B backbone with Low-Rank Adaptation (LoRA), freezing the base weights and training only lightweight adapters on all attention and MLP projection matrices, which keeps adaptation parameter- and memory-efficient. We optimize the hybrid ordinal regression loss of Eq. (5) with the paged AdamW 8-bit optimizer under a linear learning-rate schedule, training across 8 GPUs with DeepSpeed ZeRO stage 2. Table 1 summarizes the configuration.

Table 1
Fine-tuning configuration for the Qwen3-8B SLM ranker.

Component	Configuration
Backbone	Qwen3-8B
Adaptation	LoRA (r=16, α=32, dropout 0.05) on all attention and MLP projections
Optimizer	paged AdamW 8-bit, weight decay 0.01
Learning rate	1×10^{-5}, linear schedule with warmup
Effective batch	512 (4 per device $\times$ 16 grad. accum. $\times$ 8 GPUs)
Hardware	8 GPUs, DeepSpeed ZeRO-2

4. Offline Experiments and Results

In this section, we conduct an offline evaluation to assess the efficacy of our proposed learned ordinal relevance framework. We evaluate on a dataset of job–jobseeker pairs carrying the overall relevance labels of Section 3.2, retaining the full candidate list from the retrieval stage for each job. We use a time-based split in which training and validation precede the held-out test period, mirroring how production systems are deployed on future, unseen traffic and avoiding temporal leakage. Table 2 lists every system we compare and the data it was trained on; we use these names throughout.

4.1. Experimental Setup and Evaluation Metrics

To evaluate the ranking performance of the candidate sourcing models, we adopt the following standard metrics:

- **Spearman's Rank Correlation (ρ_{global}):** Measures the global monotonic relationship between the predicted rankings and the reference overall relevance labels.
- **Normalized Discounted Cumulative Gain (NDCG@k, $k \in \{5, 10, 20\}$):** Evaluates the position-aware relevance quality of the top-k ranked candidates.
- **Recall@k ($k \in \{5, 10, 20\}$):** Measures the proportion of highly relevant candidates successfully retrieved within the top-k positions.
- **High-relevance Rate@k:** Proportion of the top-k candidates that are good matches. We compute this for both jobseeker-side and employer-side relevance labels; each label is defined on a 1–5 ordinal scale, where scores of 4 and 5 are classified as highly relevant.
- **Low-relevance Rate@k:** Proportion of the top-k candidates that are poor matches. We compute this for both jobseeker-side and employer-side relevance labels; each label is defined on a 1–5 ordinal scale, where scores of 1 and 2 are classified as low-relevance.
- **Pairwise Accuracy:** Evaluates the model's capability to correctly identify the superior candidate within a given pair across distinct ordinal overall relevance classes.

Table 2
Systems compared in this paper. All fine-tuned variants optimize the hybrid ordinal regression loss of Eq. (5). Every system is scored by the expected-value read-out ("ER") of Eq. (2) over the grade tokens, except the heuristic baseline, which uses a weighted linear sum, and GPT-5.4, which is scored from open-ended generation (†). The peak-score alternative $p(5)$ is evaluated as an ablation in Section 4.6 rather than as a separate system.

System	Base model	Adaptation / data	Used in
Heuristic baseline	—	hand-tuned weights	4.2, 4.5, 5.1, 5.2
GPT-5.4 (zero-shot)†	GPT-5.4	none	4.2
GPT-4.1-mini (zero-shot)	GPT-4.1-mini	none	4.2, 4.3
GPT-4.1-mini-ft (20k)	GPT-4.1-mini	SFT, 20k pairs	4.2, 4.3
GPT-4.1-mini-ft (50k)	GPT-4.1-mini	SFT, 50k pairs	4.3, 4.4, 5.1, 7.2
Qwen3-8B-ft (50k)	Qwen3-8B	LoRA SFT, 50k pairs	4.4, 7.2
Qwen3-8B SFT (200k) *(deployed)*	Qwen3-8B	LoRA SFT, 200k pairs	4.4–4.7, 5.1, 5.2, 6

†GPT-5.4 does not expose logit_bias or constrained logprobs, so it cannot be scored by the expected-value read-out and is instead scored from open-ended generation. Its numbers are therefore not directly comparable to the other zero-shot entry.

4.2. Performance Comparison: Heuristic Baseline vs. Zero-Shot vs. Fine-Tuning

In Figure 2, zero-shot LLM configurations consistently outperform the heuristic baseline. This confirms that leveraging the pre-trained knowledge of generative models yields better text alignment than rigid, additive scoring methods.

Our fine-tuned model (GPT-4.1-mini-ft-20k) further improves over the zero-shot alternatives, achieving a 34.9% improvement in global Spearman correlation over GPT-4.1-mini (Zero-shot).

Note on Model Constraints. GPT-5.4 (Zero-shot) displays a slightly lower NDCG than GPT-4.1-mini (Zero-shot). This disparity is primarily attributed to API limitations: GPT-5.4 lacks support for logit_bias and constrained logprobs parameterization. Consequently, it is restricted to open-ended token generation, preventing the precise mathematical-expectation modeling utilized by GPT-4.1 and introducing output volatility.

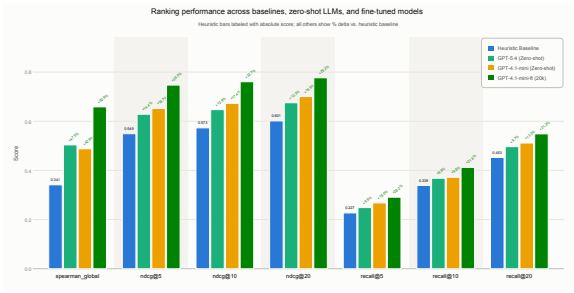

Figure 2: Performance comparison across baselines, zero-shot LLMs, and fine-tuned models. Metrics: Spearman's ρ, NDCG@$\{5, 10, 20\}$, and Recall@$\{5, 10, 20\}$. Zero-shot LLMs already surpass the heuristic, and fine-tuning adds a further consistent gain across every metric (e.g., $+34.9\%$ in global Spearman over GPT-4.1-mini zero-shot). GPT-5.4 is scored from open-ended generation rather than by the expected-value read-out, as its API exposes neither `logit_bias` nor constrained `logprobs`; its bars are therefore not a like-for-like measure of model capability.

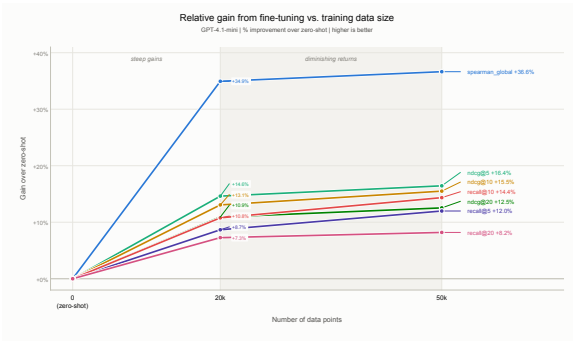

Figure 3: Performance dynamics for GPT-4.1-mini under varying fine-tuning data sizes. Most of this backbone's ranking quality is reached by ∼20k labels, with only incremental gains from 20k to 50k.

4.3. Impact of Fine-Tuning Data Scale

To explore the data efficiency and scaling behaviors of our framework, we evaluate the ranking performance across varying sizes of the annotated training dataset.

The results in Figure 3 indicate that expanding the fine-tuning corpus from 20k to 50k instances delivers gains across all ranking and recall dimensions. However, the magnitude of improvement is considerably less pronounced than the transition from zero-shot to supervised fine-tuning. Under the evaluated configuration, this suggests that meaningful task adaptation can be achieved with tens of thousands of relevance labels, even in the absence of large-scale behavioral interaction logs; for this backbone, additional labeled data continues to improve ranking quality but with diminishing marginal gains. Section 7 shows that the point at which returns diminish is itself backbone-dependent.

4.4. Proprietary LLM vs. On-Premise SLM Deployment Trade-offs

For real-world deployment, operational variables such as serving costs, API deprecation risks, and inference latency are critical. We therefore evaluate a locally hosted open-source alternative, Qwen3-8B, against the proprietary GPT-4.1-mini-ft. To separate the effect of the backbone from the effect of training-set size, we first fine-tune both on the same 50k pairs.

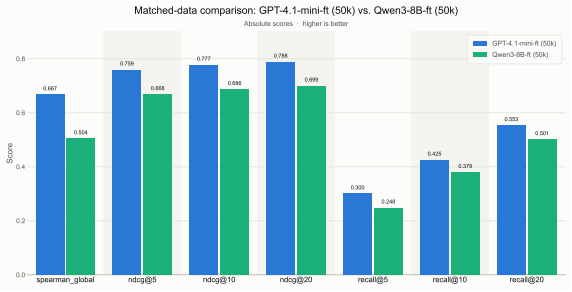

Figure 4: Matched-data comparison of the open-source SLM and the fine-tuned proprietary LLM, both trained on 50k pairs. Holding the training set fixed isolates the effect of the backbone: GPT-4.1-mini-ft leads on every metric, by $+10.4\%$ to $+32.3\%$, and most widely on global Spearman (0.667 vs. 0.504).

As shown in Figure 4, at matched training-set size GPT-4.1-mini-ft leads Qwen3-8B-ft on every metric, by $+10.4\%$ to $+32.3\%$, with the widest margin on global Spearman. Because both models see identical supervision, the gap is attributable to the backbone: differences in pre-training scale and composition, parameter count, and architecture. Scaling Qwen3-8B to 200k pairs recovers much of the difference, improving on its own 50k configuration by $+5.8\%$ to $+21.2\%$ (Table 3, hybrid column), but leaves it 4.2% to 8.4% behind GPT-4.1-mini-ft at 50k. Additional task data narrows the gap without closing it.

However, from an engineering and production standpoint, Qwen3-8B SFT maintains solid metrics that clearly outperform the legacy heuristic. Given its advantages in cost efficiency, full control over serving latency, and mitigation of upstream commercial API deprecation risks, Qwen3-8B SFT was the preferred choice for production deployment.

4.5. Pairwise Accuracy

To examine the fine-grained discrimination of our deployment model, we analyze the pairwise accuracy across different combinations of the 1-to-5 ordinal relevance labels.

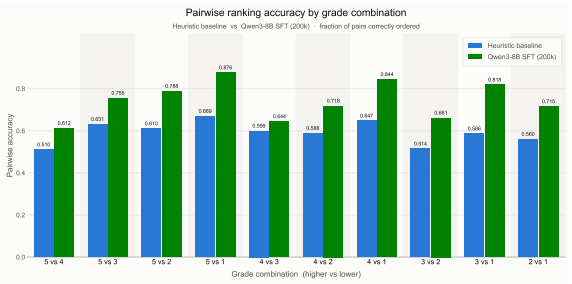

Figure 5: Pairwise accuracy breakdowns across ordinal score combinations. The model separates distant grades sharply (87.64% on 5 vs 1) while adjacent tiers (5 vs 4, 4 vs 3) remain hardest; on adjacent pairs it stays above ∼61% vs. the heuristic's near-random ∼51%, showing the ordinal loss preserves grade order.

Figure 5 shows that Qwen3-8B SFT achieves higher pairwise ranking accuracy than the heuristic baseline across all grade combinations. Performance is strongest for widely separated grades; for example, accuracy reaches 87.64% for the 5-versus-1 comparison. In contrast, adjacent grades,

such as 5 versus 4 and 4 versus 3, remain more difficult to distinguish because the differences between neighboring relevance levels are more subtle. Overall, this pattern suggests that the fine-tuned model better captures the ordinal structure of the relevance labels, while fine-grained distinctions between adjacent grades remain challenging.

4.6. Peak Score vs. Expected Value

We consider two formulas for collapsing the model's predicted distribution over the grade tokens into a single ranking score. The first, which we call the *Peak Score*, discards the full distribution and ranks by the probability mass assigned to the top grade $p(5)$ — effectively asking how confident the model is that a candidate is a top-tier match. The second is the *Expected Relevance* (ER) of Eq. (2), which aggregates the entire 1–5 distribution into a single continuous expectation. To decide which formula to serve, we recompute the pairwise accuracy under both scores across the same ordinal combinations.

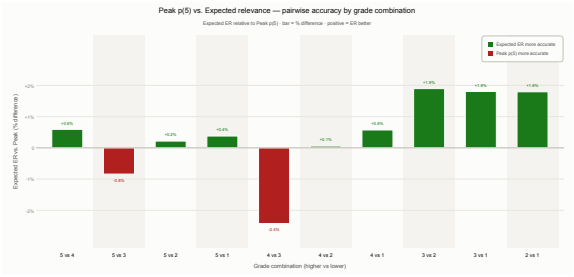

Figure 6: Pairwise accuracy difference under Peak Score $p(5)$ versus Expected Relevance (ER) scoring. ER is on par or better on most pairs and wins overall (0.7353 vs. 0.7306), with its largest gains on harder low/mid-band separations; we therefore serve ER.

As shown in Figure 6, ER edges out the Peak Score overall (0.7353 vs. 0.7306, a relative gain of $+0.64\%$) and wins the majority of individual combinations. The Peak Score holds a narrow advantage only on two pairs (**5 vs 3** and **4 vs 3**), whereas ER delivers its clearest gains on the harder low- and mid-band separations (e.g., **3 vs 1** at $+1.81\%$ and **2 vs 1** at $+1.79\%$), where collapsing the distribution to a single top-grade probability throws away discriminative signal. Because ER performs better overall and on eight of the ten grade-pair comparisons, while preserving information from the full ordinal distribution, we adopt Expected Relevance as the production ranking score. The accuracy figures reported in Figure 5 correspond to this ER scoring.

4.7. Loss-Weighting Ablation

The hybrid objective in Eq. (5) blends an ordinal-distance term (MSE) with a class-discrimination term (CE). To understand the contribution of each component, we compare our production weighting against the two single-objective extremes, holding the model, training data (200k), and all other hyper-parameters fixed:

- **MSE-only** ($w_{\text{MSE}} = 1.0$, $w_{\text{CE}} = 0.0$): optimizes solely for expected-value distance, discarding the explicit categorical objective.
- **CE-only** ($w_{\text{MSE}} = 0.0$, $w_{\text{CE}} = 1.0$): optimizes solely for grade classification, discarding the ordinal distance penalty.

Table 3
Loss-weighting ablation for Qwen3-8B SFT (200k), with 95% confidence intervals in parentheses; for MAE lower is better. No best value is marked, because the intervals overlap on all seven ranking metrics. Only exact-match accuracy and adjacent accuracy separate, and both isolate MSE-only.

Metric	MSE-only (1.0 / 0.0)	CE-only (0.0 / 1.0)	Hybrid (ours) (0.6 / 0.4)
accuracy	0.3420 (0.3180, 0.3665)	0.5044 (0.4817, 0.5264)	0.5083 (0.4863, 0.5299)
adjacent accuracy	0.7906 (0.7714, 0.8098)	0.7414 (0.7211, 0.7614)	0.7556 (0.7364, 0.7745)
mae (↓)	0.9375 (0.8947, 0.9805)	0.9602 (0.9181, 1.0017)	0.9359 (0.8933, 0.9786)
spearman_global	0.6097 (0.5759, 0.6413)	0.5963 (0.5593, 0.6312)	0.6106 (0.5754, 0.6435)
ndcg@5	0.7187 (0.6846, 0.7519)	0.7097 (0.6754, 0.7442)	0.7175 (0.6834, 0.7511)
ndcg@10	0.7349 (0.7059, 0.7641)	0.7301 (0.7009, 0.7594)	0.7324 (0.7023, 0.7619)
ndcg@20	0.7472 (0.7218, 0.7721)	0.7437 (0.7181, 0.7691)	0.7479 (0.7226, 0.7727)
recall@5	0.2803 (0.2368, 0.3275)	0.2668 (0.2246, 0.3129)	0.2830 (0.2389, 0.3303)
recall@10	0.4076 (0.3575, 0.4607)	0.4016 (0.3521, 0.4547)	0.4071 (0.3571, 0.4602)
recall@20	0.5286 (0.4788, 0.5789)	0.5267 (0.4766, 0.5768)	0.5298 (0.4804, 0.5800)

- **Hybrid (ours)** ($w_{\text{MSE}} = 0.6$, $w_{\text{CE}} = 0.4$): the deployed configuration.

To characterize the trade-off from both a classification and a ranking angle, we augment the ranking metrics of Section 4.1 with three grade-level measures:

- **Accuracy:** the exact-match rate — the fraction of candidates whose predicted grade equals the ground-truth ordinal label.
- **Adjacent Accuracy:** a relaxed variant that counts a prediction as correct when it falls within one level (± 1) of the ground-truth grade, tolerating near-miss ordinal errors.
- **Mean Absolute Error (MAE):** the average absolute difference between the predicted and true grades (lower is better), a magnitude-aware complement to the rank-based metrics.

Table 3 isolates the effect of each loss term. The ranking differences among the three objectives are small relative to their uncertainty. Across all NDCG and Recall cutoffs the 95% confidence intervals substantially overlap, providing no clear evidence that one loss consistently dominates the others in ranking quality. The point estimates do show a mild and consistent advantage for the hybrid objective on recall: it achieves the highest Recall@5 and Recall@20, while remaining effectively tied with MSE-only at Recall@10.

The more pronounced differences appear in grade-level behavior. MSE-only achieves strong adjacent accuracy but suffers a substantial drop in exact-match accuracy, suggesting that optimizing ordinal distance alone tends to produce predictions close to, but not exactly at, the target grade. CE-only restores exact-match accuracy, but yields less favorable MAE and Spearman results. The hybrid objective provides the most balanced operating point: it matches CE-only on exact classification accuracy while retaining ranking and recall performance comparable to MSE-only.

Because our downstream application places particular emphasis on recall, and the hybrid objective provides the

strongest or near-strongest recall point estimates without the exact-classification weakness of MSE-only, we retain $w_{\text{MSE}} = 0.6$ and $w_{\text{CE}} = 0.4$ for production. The uncertainty estimates suggest that this choice should be interpreted as a robustness and trade-off decision rather than as evidence of a statistically significant ranking improvement.

5. Simulation and Online Experiment Results

5.1. End-to-End Simulation

Beyond the offline metrics, we ran an end-to-end simulation that reproduces the full sourcing-agent workflow, letting us observe how the ranker behaves in context.

In this evaluation, candidate pools are dynamically generated via the upstream retrieval layer and capped at a maximum of $N = 30$ candidates per job post to match real-world screening constraints. We deploy our on-premise Qwen3-8B SFT (200k) model alongside the proprietary GPT-4.1-mini-ft (50k) model to serve inference within the scoring stage.

Experimental Control. In the production environment's legacy architecture, candidate sequences are passed to a downstream *listwise reflection* module, which applies LLM-based global heuristics and often permutes the final sequence. However, listwise reranking can introduce stochastic noise and structural distortions that mask the true sorting accuracy of the underlying point-wise ranker. To isolate the direct impact of our learned ordinal relevance and ensure a rigorous, fair empirical comparison, we bypassed the listwise ranking layer. The final ranking presentation explicitly preserves the deterministic order derived from our continuous expected-value scores.

End-to-End Simulation Metrics. We evaluate the simulation results across two primary dimensions: Employer Relevance (alignment with job descriptions) and Jobseeker Relevance (alignment with candidate preferences and historical match success). Performance gains are calculated as relative improvements over the legacy heuristic baseline.

Table 4
Relative performance lift (%) in the end-to-end simulated sourcing pipeline.

Evaluation Dimension & Metric	Qwen3-8B SFT (200k)	GPT-4.1-mini ft (50k)
Employer Relevance NDCG@10	$+3.1\%$	$+5.9\%$
Employer Relevance NDCG@20	$+2.6\%$	$+4.2\%$
Employer Relevance High Relevance Rate@20	$+4.5\%$	$+6.2\%$
Employer Relevance Low Relevance Rate@20	-23.8%	-27.3%
Jobseeker Relevance NDCG@10	$+54.2\%$	$+63.7\%$
Jobseeker Relevance NDCG@20	$+36.7\%$	$+42.4\%$
Jobseeker Relevance High Relevance Rate@20	$+32.7\%$	$+37.7\%$
Jobseeker Relevance Low Relevance Rate@20	-46.7%	-56.2%

The legacy heuristic baseline is, by construction, a one-sided "employer hard-requirements checklist" hand-crafted from the job description (JD). It therefore already encodes employer-side preferences reasonably well and sits near a local optimum on that dimension, while leaving jobseeker-side alignment essentially unoptimized. The simulation lifts follow this asymmetry directly: Employer Relevance improves moderately (NDCG@10 of $+3.1\%$ / $+5.9\%$), whereas Jobseeker Relevance, which the baseline never targeted, rises far more ($+54.2\%$ / $+63.7\%$). The large jobseeker-side gains therefore measure headroom in the baseline rather than an order-of-magnitude absolute advantage for the ranker.

5.2. Online Experiment

To evaluate the real-world utility of the proposed ranker, we conducted a live online evaluation. We randomized at the employer level rather than the account level: an employer may operate several accounts, so randomizing by employer keeps the ranking experience consistent across all of an employer's accounts and prevents any single employer from being exposed to both arms.

1. **Control:** the production heuristic baseline of Section 3.1 (per-criterion LLM judgments aggregated by a hand-tuned weighted sum).
2. **Treatment:** the single-token expected-value score from the fine-tuned SLM (Section 3.2).

Table 5
Online A/B results: relative lift of treatment (fine-tuned SLM) over control (heuristic), with 95% confidence intervals in parentheses.

Metric	% gain wrt control
Employer High-relevance Rate (↑)	$+5.17\%$ ($+3.5\%$, $+6.8\%$)
Employer Low-relevance Rate (↓)	-27.31% (-35%, -20%)
Jobseeker High-relevance Rate (↑)	$+33.79\%$ ($+29.9\%$, $+37.7\%$)
Jobseeker Low-relevance Rate (↓)	-19.73% (-27%, -13%)
Apply Rate (↑)	$+5.68\%$ (0%, $+11.0\%$)
Keep / (Keep+Remove) Rate (↑)	$+7.07\%$ ($+5.0\%$, $+9.1\%$)

We report the relative lift of treatment over control (Table 5), with 95% confidence intervals. We measure success along the same two relevance axes used offline — Employer Relevance and Jobseeker Relevance, each reported as a high- and low-relevance rate. We also track two behavioral signals: the keep rate, the fraction of surfaced candidates an employer keeps rather than removes during review (a direct employer signal), and the apply rate (a jobseeker signal).

The online numbers line up with what we saw offline and in simulation. Swapping the additive heuristic for the learned single-token ranker improved relevance for both employers and jobseekers, and it cut the share of bad matches shown to employers by 27.3%. The keep-rate result matters most to us: it comes from employers actually keeping or removing surfaced candidates, not from model-generated labels, so the 7.07% gain is a sign the improvement is real and not an artifact of evaluating against our own labeler. Apply rate moved in the right direction as well, rising 5.68%, which is directionally positive.

6. System Architecture & Engineering Implementation

The proposed ordinal ranker is deployed as the second-stage point-wise scoring module within a multi-stage production pipeline, upstream of the separate listwise reranking stage. Decoupling scoring from retrieval confines heavy language

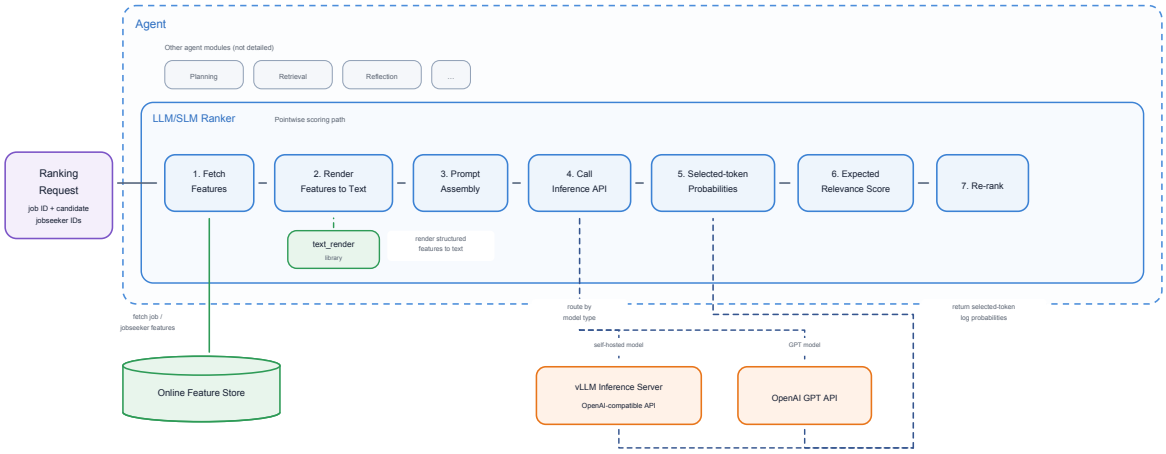

Figure 7: Serving architecture. At inference, candidate and job features are fetched from an online feature store and assembled into an LLM-ready prompt by a text-rendering step. Depending on the variant, scoring is served either by a vLLM inference server (Qwen3-8B with per-variant LoRA adapters) or by the OpenAI API; in both cases the model produces a distribution over the grade tokens $\{1, 2, 3, 4, 5\}$, from which we compute the expected value (ER) and rank candidates by it.

model inference to a refined candidate subset, keeping per-session latency low without sacrificing ranking quality. The remainder of this section details the serving architecture (Figure 7), data pipeline, latency optimizations, production infrastructure, and operational monitoring.

6.1. Data Pipeline and Consistency

To mitigate training-serving skew, we employ a versioned rendering configuration shared by training and serving. This ensures identical feature hydration across offline training and online inference. The rendered model input combines job-related information, candidate-related information, matching constraints, and other task-relevant contextual signals.

6.2. Inference Latency Optimization

To reduce inference latency, we implement three optimizations:

1. **Concurrent Scoring**: We utilize asynchronous I/O with shared connection pools to enable concurrent per-candidate scoring, avoiding request serialization.
2. **Feature-Fetch Overlapping**: Candidate metadata retrieval is executed in parallel with upstream matching logic, effectively masking network round-trip latency.
3. **Constrained Decoding**: By treating the task as a single-token classification, we remove the computational overhead of variable-length decoding and post-hoc output parsing.

6.3. Production Infrastructure

The system is deployed on on-premise Kubernetes clusters as a vLLM-based service, utilizing a shared foundation model (Qwen3-8B) with per-variant LoRA adapters. This modular architecture supports multi-tenant serving, where multiple ranking objectives share the same GPU memory footprint.

We optimize inference using fp8 precision for both weights and KV-cache, alongside continuous batching and prefix caching to minimize recomputation of the shared system prompt (Table 6). Capacity scales horizontally with demand to sustain throughput under bursty, per-session fan-out patterns.

Table 6
vLLM serving and inference parameters.

Component	Configuration
Base / adapter	Qwen3-8B base + per-variant LoRA adapter
Precision	fp8 weights, fp8 KV cache
Context length	16,384 tokens
Runtime	vLLM 0.22.0, self-hosted

6.4. Operational Monitoring

We maintain stability through layered observability across three planes: serving infrastructure, model quality, and service-level reliability.

On reliability, each scoring call is tagged with a mutually exclusive outcome — scored, no_logprobs, parse_reject, or call_error. Since any non-scored call degrades gracefully to the deterministic heuristic, this taxonomy separates infrastructure failure from silent quality loss. From it we derive three production SLOs over a rolling 7-day window: *scoring availability* (non-error rate; target 99%, observed ≈99.95%), *usable-score rate* (the stricter fraction of calls yielding a usable score; target 99%, observed ≈99.8%, whose gap to availability isolates silent degradation), and *per-session stage latency* (p95 $\leq$ 8 s, measured at the per-session stage rather than the per-candidate scoring call of Section 6.2). Each objective is guarded by multi-window burn-rate alerts, pairing a fast window for acute outages with a slow window for gradual budget erosion, and is defined as version-controlled SLO-as-code alongside per-variant diagnostic monitors for

triage.

Beyond standard infrastructure telemetry, we monitor the distribution of ER scores and token logits to detect training/serving drift. Proposed ranking models undergo a 48-hour shadow validation phase, auditing performance against legacy baselines on live traffic prior to production promotion.

7. Analysis & Discussion

7.1. Fine-Grained Discrimination and Ordinal Consistency

The pairwise breakdown in Figure 5 reveals a clear pattern across the 1-to-5 ordinal labels, and it shows how well the deployed model preserves numerical distance relative to the heuristic baseline.

The model separates distant grades sharply — 87.64% pairwise accuracy on the "5 vs 1" combination, which distinguishes highly qualified candidates from entirely unqualified ones.

By contrast, distinguishing adjacent tiers such as "5 vs 4" and "4 vs 3" remains substantially harder, with accuracy in the range of roughly 61%–64%. This is consistent with the realities of recruitment, where even human expert annotators face difficulty separating them. These borderline qualifications occupy an inherently grey area, and the residual error largely reflects this annotation ambiguity rather than a failure of the model.

On adjacent combinations, the legacy heuristic performs close to random guessing (approximately 51%), whereas Qwen3-8B SFT sustains accuracy above 61%. The model thus preserves the ordering between neighboring grades far better than the heuristic, rather than treating the labels as unordered classes.

7.2. Scaling Laws vs. Foundation Capacity Trade-offs

Two figures bear on the trade-off between data efficiency and foundation-model capacity: Figure 3 (GPT-4.1-mini at 20k vs. 50k) and Figure 4 (GPT-4.1-mini and Qwen3-8B at a matched 50k).

As shown in Figure 3, the gain from 20k to 50k samples is small and quickly flattens for GPT-4.1-mini: that backbone reaches most of its ranking quality by 20k labels. Data efficiency of this kind matters for cold-start, low-traffic settings, where the large interaction logs that deep neural rankers require are unavailable. It is, however, a property of the backbone rather than of the task: Qwen3-8B is still improving well past that point, gaining $+5.8\%$ to $+21.2\%$ from 50k to 200k pairs.

Figure 4 isolates the complementary limitation. With training data held fixed at 50k, GPT-4.1-mini-ft leads Qwen3-8B-ft by $+10.4\%$ to $+32.3\%$ across the seven metrics, so the difference is attributable to the backbone rather than to supervision. Scaling Qwen3-8B to 200k narrows but does not close it: with four times the training data the deployed model remains 4.2% to 8.4% behind GPT-4.1-mini-ft at 50k. Beyond a certain point the bottleneck therefore shifts from data volume to the representational capacity of the base model itself.

7.3. Determinism: Variance Reduction and Latency Optimization

Error cascades in long generation paths. Conventional two-stage approaches — first emitting a complex JSON structure or extracted reasoning, then producing a score — require longer generated output. Because the output is autoregressive, a small probability deviation in an early token can be amplified in a cascade along the decoding chain (an "error cascade"), injecting variance into the final score.

Constrained single-token decoding. Our approach instead extracts the logits at the very first generated token, computes the expected value, and terminates the interaction immediately. Because only a single token is decoded, there are no earlier tokens whose small numerical fluctuations could cascade and be amplified into a large deviation in the final score. In addition, cross-entropy fine-tuning drives the model toward confident, low-entropy grade distributions [13, 14], so the small floating-point nondeterminism of GPU inference rarely changes the selected grade and only negligibly perturbs the expected-value score. Together these remove the sampling and generation-cascade sources of variance and greatly reduce the model's per-call scoring variance, while also cutting decoding overhead.

8. Summary and Future Work

8.1. Conclusion

For cold-start and data-scarce recommendation settings, this work narrows the gap between deep neural rankers that depend on large click logs and hard-coded, linearly weighted heuristics. In their place, we present an approach for fine-grained text alignment based on SLM ordinal regression.

The proposed hybrid ordinal regression loss (MSE + CE), combined with single-token expected-value scoring, preserves numerical monotonicity: on adjacent grade pairs the deployed model stays above 61% pairwise accuracy where the heuristic is close to random. Alongside this, we observe strong screening of high-value candidates and a sharp reduction in low relevance rate (-46.7%).

We show that a locally deployed SLM can remove the generation variance of open-ended decoding and meet stringent online latency requirements, while still drawing on the world knowledge embedded in the underlying model.

The online experiment provides further evidence that the improvements observed offline and in simulation are also realized in the production environment. Improvements in model-assessed relevance were accompanied by stronger employer interaction behavior, suggesting that the learned ranker is surfacing quality candidates to employers.

Together, these results offer a practical recipe for building talent-sourcing rankers under data scarcity.

8.2. Future Work

From point-wise to list-wise fine-tuning. The current framework scores each candidate independently for a single job. We plan to introduce list-wise objectives into the fine-tuning stage so that the model learns not only to score individuals in isolation but also to reason about the relative ordering among candidates within a job.

Further latency reduction. We will continue to optimize inference latency to push the system toward even lower-cost, higher-throughput production serving.

Declaration on Generative AI

During the preparation of this work, we used ChatGPT and Claude as writing and formatting aids, with every output reviewed and edited by us. Specifically, the tools were used to: check grammar and spelling; paraphrase and reword; improve writing style - sentence structure, word choice, and flow; create charts and diagrams from our data and experimental results; assist with citation management and latex formatting; and refine abstract. All research design, data, results, and conclusions are ours. No text or figures were used without our revision, and we take full responsibility for the publication's content.

References

[1] S. M. U. Dadaboyev, J. Abdullayeva, N. Abbosova, A. Suleymenova, K. Mamadjanova, Role of artificial intelligence in employee recruitment: systematic review and future research directions, Discover Global Society (2025).

[2] X. Han, C. Zhu, X. Hu, C. Qin, X. Zhao, H. Zhu, Adapting job recommendations to user preference drift with behavioral-semantic fusion learning, in: ACM SIGKDD, 2024.

[3] X. Yu, R. Xu, C. Xue, J. Zhang, X. Ma, Z. Yu, ConFit v2: improving resume-job matching using hypothetical resume embedding and runner-up hard-negative mining, in: Findings of ACL, 2025.

[4] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, H. Xiong, Enhancing person-job fit for talent recruitment: an ability-aware neural network approach, in: SIGIR, 2018.

[5] S. Guo, F. Alamudun, T. Hammond, RésuMatcher: a personalized résumé–job matching system, Expert Systems with Applications (2016).

[6] Y. Liang, L. Yang, C. Wang, X. Xu, P. S. Yu, K. Shu, Taxonomy-guided zero-shot recommendations with LLMs, in: COLING, 2025.

[7] H. Zhuang, Z. Qin, K. Hui, J. Wu, L. Yan, X. Wang, M. Bendersky, Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels, in: NAACL, 2024.

[8] S. Ouyang, J. M. Zhang, M. Harman, M. Wang, An empirical study of the non-determinism of ChatGPT in code generation, ACM Transactions on Software Engineering and Methodology (2025).

[9] P. Liu, J. Shen, Q. Shen, C. Yao, K. Kao, D. Xu, R. Arora, B. Zheng, C. Johnson, L. Hong, J. Wu, W. Zhang, Powering job search at scale: LLM-enhanced query understanding in job matching systems, in: CIKM, 2025.

[10] F. Guo, W. Li, H. Zhuang, Y. Luo, Y. Li, L. Yan, Q. Zhu, Y. Zhang, MCRanker: generating diverse criteria on-the-fly to improve pointwise LLM rankers, in: WSDM, 2025.

[11] L. Shi, C. Ma, W. Liang, X. Diao, W. Ma, S. Vosoughi, Judging the judges: A systematic study of position bias in LLM-as-a-judge, in: IJCNLP-AACL, 2025.

[12] Y. Pei, Y. W. Pang, W. Cai, N. Sengupta, D. Toshniwal, Leveraging LLM generated labels to reduce bad matches in job recommendations, in: RecSys, 2024.

[13] Z. Li, C. Chen, T. Xu, Z. Qin, J. Xiao, Z.-Q. Luo, R. Sun, Preserving diversity in supervised fine-tuning of large language models, in: ICLR, 2025.

[14] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, G. Hinton, Regularizing neural networks by penalizing confident output distributions, in: ICLR Workshop, 2017.