Deep Sourcing: Agentic Multi-Strategy Search with Iterative Reflection for Talent Discovery

Joshua H. Levy[1,*,†], Willy Lew Yuk Vong[1,†], Jacob Plains[1,†], Ravi Kiran Divvela[2,‡], Kevin Ka'aumoana Duarte[1], Wenhui Liu[1], Manasvi Nishith Vohra[1] and Bhushan Shitole[1,†]

[1]*Indeed.com, USA*
[2]*Faire, USA*

Abstract

Traditional candidate search in talent acquisition operates as a single-pass retrieval: a recruiter formulates a query, a matching system returns ranked results, and the recruiter manually iterates if results are unsatisfactory. Single-pass retrieval is a poor fit, since the same role can be expressed through multiple valid search formulations that emphasize different skills, titles, backgrounds, or adjacent experiences. We present Deep Sourcing, a production candidate discovery system that uses large language models (LLMs) to plan multiple complementary search strategies, execute them against a large-scale retrieval platform, evaluate candidate profiles against employer-given criteria, and optionally refine later search rounds through reflection. The system combines high-recall retrieval with LLM-based semantic evaluation, but computes profile-match scores deterministically from criterion-level judgments. In production experiments at Indeed, Deep Sourcing improved Precision@10 for good matches by 35.5% and for optimal matches by 296.4% relative to a strong production baseline. A single-loop variant captured most of the precision gain for good matches, while iterative reflection improved coverage and increased the share of optimal matches. These results suggest that the main value of agentic candidate search lies in multi-strategy decomposition and structured evaluation, with reflection providing targeted incremental gains.

Keywords

Agentic search, Large language models, Candidate sourcing, Talent acquisition, LLM-as-judge, Multi-strategy retrieval, Iterative reflection, Recommender systems

1. Introduction

Candidate search in talent acquisition is a large-scale retrieval problem: given a job opening with complex, often ambiguous requirements, identify the strongest matches from a pool of candidate profiles. On Indeed, an employer has access to 370 million profiles on the marketplace[1] and to past applicants to the employer's other job postings. Modern retrieval pipelines typically combine sparse retrieval, such as BM25 [1, 2], with dense retrieval based on embedding nearest-neighbor search [3] and learned rankers [4]. Production systems often go further by combining several retrieval strategies, each optimized for a different candidate discovery pattern. At the time of our experiments, the production baseline used 10+ such strategies, with results merged, filtered, and reranked by a learned ranking model. We describe this baseline in Section 4.1 at the level needed to interpret the comparison.

This paradigm has several limitations. First, a single query encodes one interpretation of the hiring need and may miss candidate profiles reachable through alternative formulations. For example, a query for *Software Architect* may miss profiles with *Principal Software Engineer* titles. Second, the retrieval system has no mechanism to evaluate the quality of its own results and adjust accordingly. Third, when results are unsatisfactory, the recruiter must manually hypothesize why, reformulate, and re-execute the search. This process is slow, inconsistent, and difficult to scale.

These limitations motivate a different approach. Rather than single-pass retrieval, a sourcing system can reason about which search strategies to try, execute multiple strategies in parallel, evaluate whether results are satisfactory, and iteratively refine its approach in a manner similar to an experienced recruiter.

We present **Deep Sourcing**, an agentic search system that implements this approach. The system evaluates candidate profiles against structured evaluation criteria: a weighted rubric of skills, qualifications, and attributes derived from the job requirements and recruiter instructions. Drawing on advances in LLM-based planning [5], tool use [6, 7], and self-reflection [8, 9], Deep Sourcing orchestrates an iterative loop with four phases:

- **Plan**: An LLM analyzes the job requirements, recruiter instructions, and evaluation criteria to generate multiple search strategies, each targeting a different facet of the candidate profile space.
- **Execute**: Each strategy is executed against a production profile retrieval platform.
- **Evaluate**: An LLM-as-judge pipeline [10] performs pointwise profile-match evaluation against the employer-provided criteria.
- **Reflect**: The system evaluates aggregate result quality and decides whether to expand successful strategies, continue reviewing deeper results from a strategy, abandon unproductive approaches, or try new strategies. It then loops back to planning if needed.

Deployed at Indeed, the world's #1 job site,[2] Deep Sourcing operates in both a batch mode that runs autonomously in

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.
*Corresponding author.
†These authors contributed equally to this research.
‡Work performed while at Indeed.com.
joshualevy@indeed.com (J. H. Levy); willylew@indeed.com (W. L. Y. Vong); jplains@indeed.com (J. Plains); ravi.divvela@faire.com (R. K. Divvela); kduarte@indeed.com (K. K. Duarte); wliu@indeed.com (W. Liu); mvohra@indeed.com (M. N. Vohra); bhushans@indeed.com (B. Shitole)
0009-0006-1079-4053 (J. H. Levy); 0009-0008-2264-1083 (W. L. Y. Vong); 0009-0005-9566-9931 (J. Plains); 0009-0000-3140-0021 (R. K. Divvela); 0009-0002-9319-2226 (K. K. Duarte); 0009-0008-9814-5107 (W. Liu); 0009-0009-4440-4414 (M. N. Vohra); 0009-0007-4424-4591 (B. Shitole)

[1]Indeed data (worldwide), job seeker accounts set to public that have a unique, verified email address

[2]Comscore, Total Visits, March 2026

the background and an interactive mode triggered through a conversational interface with progressive result surfacing.

Contributions. (1) We describe the architecture of a production agentic search system, detailing the planning, execution, evaluation, and reflection components. (2) We present a multi-strategy planning mechanism in which an LLM decomposes complex hiring needs into complementary search strategies, including LLM-generated boolean queries. (3) We introduce a two-stage LLM evaluation pipeline combining pointwise scoring against weighted criteria groups with listwise reranking. (4) We report production results demonstrating significant improvements over a mature retrieval baseline and discuss lessons from operating at scale.

2. Related Work

Deep Sourcing draws on several research areas. Agentic retrieval-augmented generation extends traditional RAG [11, 12] by giving LLMs control over when and how to retrieve. Systems such as Self-RAG [9], CRAG [13], and Adaptive-RAG [14] dynamically adjust retrieval strategy based on result quality or query complexity; Singh et al. [15] provide a comprehensive survey. Deep Sourcing extends this paradigm from single-document retrieval to multi-strategy candidate profile search. Deep Sourcing's decomposition of a complex information need into multiple retrieval steps is similar to CoRAG [16]. However, CoRAG does so by training an LLM to iteratively reformulate a single query into a sequential retrieval chain, while Deep Sourcing uses an LLM planner to generate multiple complementary strategies executed in parallel against a production retrieval platform, without any model training.

Using LLMs as evaluators has emerged as a scalable alternative to human annotation. MT-Bench [10] demonstrated high agreement between LLM and human preferences; subsequent work has explored pointwise, pairwise, and listwise prompting strategies for ranking [17]. More recent generative-LLM reranking methods include listwise rerankers [18, 19], setwise prompting [20], attention-based zero-shot reranking [21], automatic prompt engineering for reranking [22], and aspect-aware reranking frameworks for recommendation [23]. Closest to our setting, Jouanneau et al. [24] prompt a generative LLM to assign per-candidate relevance scores on a freelancing marketplace. They then distill those scores into a lightweight reranker for real-time inference. Deep Sourcing deploys a two-stage evaluation pipeline, using LLMs for pointwise scoring against structured criteria and for listwise reranking.

Planning and reflection in LLM agents have been studied through ReAct [6], Reflexion [8], and Tree of Thoughts [5]. Deep Sourcing instantiates these principles in a domain-specific system where a planner generates search strategies, executors retrieve candidate profiles, and a judge evaluates results and triggers reflection. Lo et al. [25] present a related multi-agent architecture focused on resume screening rather than discovery. To our knowledge, Deep Sourcing is the first published production system applying agentic, multi-strategy search with iterative reflection to talent discovery.

3. Methods

3.1. Overview

Deep Sourcing is implemented as a stateful, cyclic workflow: a directed graph where nodes represent processing stages, edges represent transitions, and each stage reads from and updates a shared session state.

A session begins after the search intent has been confirmed. The system proceeds through context loading, criteria initialization, and then enters an iterative loop of planning, execution, evaluation, and reflection, illustrated in Figure 1. The loop terminates when the exit conditions are met (Section 3.8), and the results are finalized. Each LLM-driven stage is independently configurable. The models used for the experiments in this paper are specified in Section 4.1.

3.2. Criteria Initialization

Before the session begins, an LLM analyzes the job description and any additional recruiter instructions to produce a structured set of evaluation criteria. The output is a set of **weighted criteria groups**, for example:

```
CriteriaGroup:
  criteria: ["Python", "Java", "C++"]
  weight: 0.30
  aggregator: "max"
  reason: "The job description lists one of
       these programming languages as required"
```

Each group defines: (a) the skills, qualifications, or attributes in the group; (b) a weight reflecting relative importance based on specifications from the recruiter, with weights summing to 1.0 across groups; and (c) an aggregation method such as `max` when any criterion in a group suffices or `avg` when all criteria contribute to the group score. This structured rubric ensures consistent evaluation across strategies and loop iterations.

3.3. Multi-Strategy Planning

Given the context of the job, the evaluation criteria, and any reflection learnings from prior loops, the planning node generates multiple search strategies per iteration. Each strategy represents a distinct retrieval approach:

The **boolean precision query** strategy is notable because it uses an LLM to generate a structured Lucene boolean query encoding its understanding of the ideal candidate profile. For example:

```
+title:("Senior Software Engineer" OR
          "Staff Engineer")
+skill:("Python" OR "Django" OR "Flask")
+degreeType:("Bachelor" OR "Master")
```

This leverages LLM reasoning for query formulation while using inverted-index search for efficient retrieval. The planner also provides a natural-language explanation of why each strategy was chosen and how the strategies complement one another.

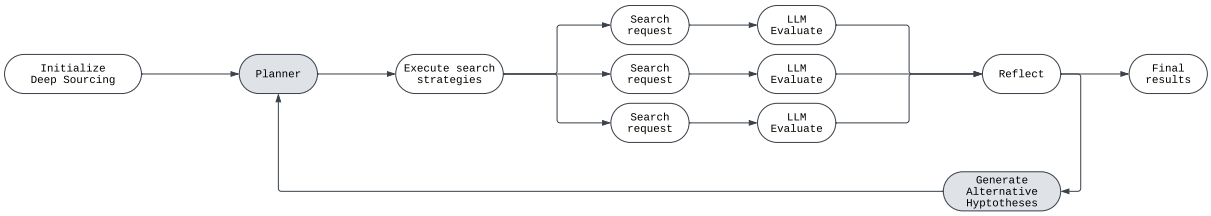

Figure 1: Deep Sourcing system architecture. Diagram of the Plan, Execute, Evaluate, and Reflect loop in Deep Sourcing.

Table 1
Search strategies used by Deep Sourcing.

Strategy Type	Description	Retrieval Method
Criteria-based search	Broad search using structured skill and experience criteria	Dense and sparse retrieval with criteria matching
Boolean precision query	Targeted search using LLM-generated Lucene boolean queries	BM25 with structured boolean queries
Applicant pool search	Search profiles belonging to candidates who have applied to this or similar jobs	BM25 with sparse vector retrieval
Recommendation-based search	Recommendations based on job-relevant skills and qualifications, surfacing similar profiles where the reported experience aligns with recent outreaches or fits the job	Collaborative filtering and learned embeddings

3.4. Execution

For each planned strategy, the execution node converts the strategy into a retrieval request, obtains a ranked list of candidate profiles, de-duplicates profiles already seen in the session, and prepares profiles for evaluation. The execution layer treats the retrieval platform as a tool: strategy-specific parameters are passed to the platform, and the platform returns candidate profiles with the information needed for downstream evaluation.

3.5. Candidate Profile Evaluation

Each retrieved candidate profile is individually evaluated by an LLM against the employer's criteria. The LLM produces:

- A **feature matrix**: per-criterion scores on a five-level scale.
- A **match explanation**: natural-language reasoning about the candidate profile's fit.

The profile-match score is computed deterministically from the feature matrix using the weighted criteria groups, rather than asking the LLM to produce a scalar score directly. This decoupling of semantic evaluation from deterministic scoring is a key design decision discussed further in Section 5.1.

3.6. Reflection and Iteration

After execution and evaluation, the reflection node receives candidate profiles with their feature matrices, the strategies that produced them, results from prior loops, and the evaluation criteria. It performs three functions:

Listwise reranking. Candidate profiles from all strategies and loops are comparatively ranked. Each profile receives a global rank and comparative reasoning, meaning a natural-language explanation of its relative positioning. Comparative reasoning follows strict guidelines: candidate identifiers are not present in its input, the explanations it generates must use gender-neutral language, and assessments are limited to professional qualifications.

High-confidence match identification. From the ranked list, the LLM identifies high-confidence matches: candidate profiles that are strongly aligned with the evaluation criteria and raise no significant concerns about match relevance.

Structured learning extraction. The LLM produces structured learnings that capture why strategies produced useful or poor results, identify which, if any, criteria were not supported by a given strategy, and specify whether additional pages from a strategy are worth evaluating. These learnings are fed back into the next planning iteration, enabling the planner to expand productive strategies, stop unproductive ones, and generate new strategies informed by accumulated evidence.

Not all evaluation criteria can be expressed as retrieval constraints for every strategy. For example, some criteria may be captured by a boolean query, while others require semantic evaluation after retrieval. The reflection node can therefore reason about whether lower-ranked results are provably unlikely to improve, whether a strategy should be paged further, or whether a new strategy is needed.

3.7. Candidate Profile Pruning

Within each loop iteration, only the top-K candidate profiles per strategy are carried forward for evaluation and reranking, where K is calibrated to the employer's target result count. For any per-strategy ranked list, at most K profiles from that list can appear in the overall top-K result set, so retaining more than K candidate profiles cannot change the final result unless de-duplication, filtering, or downstream re-scoring removes or demotes profiles retained above them. In practice, these exceptions determine the value of K. This pruning bounds LLM evaluation cost while preserving the candidate profiles most likely to affect the final ranked list (see Section 5.2).

3.8. Exit Conditions

The loop terminates when: (1) the executed loop count reaches a configurable maximum, typically 2–4, or (2) the number of high-confidence matches (Section 3.6) exceeds a target threshold. A minimum loop parameter ensures at least one full iteration before early termination is considered.

3.9. Single-Loop Variant

For latency-sensitive use cases, a **single-loop variant** executes 2+ retrieval strategies without the reflection-driven iteration, optionally using heuristic scoring instead of full weighted criteria initialization. This variant reduces end-to-end latency by approximately 80% and serves as both a product feature and a comparison baseline for evaluating the incremental value of iterative reflection.

4. Results

4.1. Experimental Setup

We evaluate Deep Sourcing against a production baseline and a single-loop variant of itself:

- **Production baseline**: Indeed's production recommendation pipeline at the time of the experiments. The baseline used 10+ retrieval strategies spanning sparse retrieval [2], dense retrieval including two-tower architectures [26], graph neural network embeddings [27], and candidate-profile-to-candidate-profile collaborative filtering [28]. Results are merged, filtered, and reranked by a production deep neural network ranker.
- **Single-loop variant** (Section 3.9): isolates the value of multi-strategy planning from the value of iterative reflection.

For the evaluations reported in this paper, Deep Sourcing is configured to run for up to 3 reflection loops, with the planner generating up to 3 search strategies per loop (see Section 3.8 for exit conditions).

The models used in Deep Sourcing are configurable and have evolved over time. The experiment reported in Section 4.2 was conducted in August 2025 using a version of Deep Sourcing that used the following combination of frontier and cost-efficient models from OpenAI: *gpt-4.1-mini* for initialization and evaluation, *gpt-4.1* for planning, and *o4-mini* for reflection. The 12-week experiment reported in Section 4.3 also began in August 2025 and used the same configuration. The token cost for these experiments was approximately 60 cents per session on average.

4.2. Match Quality Across Systems

Our offline evaluations depend on ground-truth labels that are assigned through a combination of automated LLM-based labeling and manual validation. The match between a job and a candidate profile is evaluated along several dimensions such as title, industry, career level, and qualifications using a five-point scale for each dimension. Heuristics are used to aggregate these dimensional ratings into an overall match rating on a five-point scale.

The overall match rating can be thought of as two levels of good matches, a neutral label, and two levels of poor matches. The higher of the two good labels is the **optimal** label. The primary relevance metrics are based on Precision of the top-10 candidate profiles returned by the recommendation system. **Prec(good)@10** is the fraction of the top-10 profiles that are rated with either of the good labels, while **Prec(optimal)@10** is the fraction of the top-10 that receive the optimal rating. By definition, Prec(optimal)@10 $\leq$ Prec(good)@10. The **bad-match rate** is the fraction of top-10 results that receive either of the poor ratings.

Note that the ground-truth labels discussed in this section are computed offline, using different LLM instructions and a different model[3] than those used in-loop. This is done to limit self-preference bias [29]. While the high-confidence match label (Section 3.6) is primarily used internally during the Reflection stage, we do track the number of high-confidence matches as a secondary relevance metric.

We compare Deep Sourcing with the single-loop variant and production baseline using an offline evaluation of 100 Deep Sourcing sessions. Note that top results from all three methods are computed and logged during each Deep Sourcing run.

Table 2 summarizes match quality across systems. All values are reported as relative improvements over the production baseline, with bootstrapped 95% confidence intervals.

Deep Sourcing achieves statistically significant gains relative to the production baseline: a 35.5% improvement in Prec(good)@10 and a 296.4% improvement in Prec(optimal)@10. Bad-match rate also improves (7.9% decrease), although this improvement is not statistically significant. Consequently, the extent to which the additional good matches replace poor matches vs. neutral matches is unclear.

The single-loop variant accounts for much of the Prec(good)@10 improvement and a significant Bad-match rate decrease, showing that multi-strategy planning alone provides substantial value. Iterative reflection contributes in two further ways. First, it upgrades the quality of matches within the top-10: although Deep Sourcing adds only modestly to the Prec(good)@10 lift, it improves Prec(optimal)@10 by 70.5% [+46.4%, +100.0%] over the single-loop variant. Second, it improves **match coverage** by finding candidate profiles that the single-loop variant does not surface, as ~34% of the good matches identified by Deep Sourcing are found in its second or third loop.

We report match quality metrics for the top-five occupation categories plus a residual catch-all in Table 3. Deep Sourcing shows significant increases in Prec(good)@10 and Prec(optimal)@10 for all categories where the lift is defined. Interestingly, Deep Sourcing increases the Bad-match rate for the Sales & Retail category, though the confidence interval crosses zero. By contrast, Deep Sourcing reduces the Bad-match rate below the production baseline and the single-loop variant in the Technology, Protective & Security, and Finance & Accounting categories. Excluding Sales & Retail, Deep Sourcing reduces the Bad-match rate by 34.6% [-45.4%, -22.3%] versus the production baseline. The Sales & Retail category is unique because the single-loop variant outperforms Deep Sourcing on the Prec(good)@10 metric. Deep Sourcing is more effective than the single-loop variant for the Prec(optimal)@10 metric in Sales & Retail, and for both Prec(good)@10 and Prec(optimal)@10 in the non-sales

[3] OpenAI's *o3-mini* for this experiment

Table 2
Match quality comparison. Prec(good)@10 measures the fraction of top-10 candidate profiles labeled with either of the two good overall ratings. Prec(optimal)@10 measures the fraction receiving the optimal label. Bad-match rate measures the fraction receiving either of the two poor-quality labels. Values are shown as relative lift over the production baseline with bootstrapped 95% confidence intervals.

System	Prec(good)@10	Prec(optimal)@10	Bad-match rate
Single-loop variant	+29.1% [19.6, 39.7]	+132.5% [82.4, 201.6]	-18.9% [-27.9, -8.9]
Deep Sourcing (iterative)	+35.5% [25.9, 45.9]	+296.4% [220.4, 408.6]	-7.9% [-17.1, +2.5]

Table 3
Match quality comparison for the top-five occupations in our evaluation set and a catch-all (Other). **Bold** cells indicate a 95% CI that lies entirely outside the corresponding CI reported in Table 2. Italicized cells are discussed in greater detail in the prose. Values are shown as relative lift over the production baseline with bootstrapped 95% confidence intervals.

Occupation	System	Prec(good)@10	Prec(optimal)@10	Bad-match rate
Sales & Retail	Single-loop variant	+29.9% [+16.8, +44.3]	+134.1% [+61.4, +260.1]	-9.0% [-22.8, +6.5]
Sales & Retail	Deep Sourcing (iterative)	+15.5% [+4.3, +28.5]	+203.4% [+115.9, +361.7]	*+14.4% [-0.3, +32.1]*
Technology	Single-loop variant	+18.3% [+4.0, +34.4]	+100.5% [+46.9, +183.7]	-26.8% [-48.1, +2.3]
Technology	Deep Sourcing (iterative)	+40.9% [+26.0, +58.5]	+235.3% [+153.3, +359.6]	**-51.6% [-67.9, -29.4]**
Business Ops & Management	Single-loop variant	+36.6% [+2.4, +85.0]	+412.0% [-31.7, +821.7]	-48.8% [-76.7, -4.9]
Business Ops & Management	Deep Sourcing (iterative)	+57.4% [+22.4, +106.6]	+1788.9% [+372.2, +2450.0]	-18.4% [-54.5, +39.5]
Protective & Security	Single-loop variant	+35.4% [-9.6, +105.3]	*+21.9% [-100, +387.5]*	-61.0% [-84.8, -25.5]
Protective & Security	Deep Sourcing (iterative)	**+163.6% [+100.4, +279.7]**	**+1850% [+602, +4385]**	**-84.4% [-96.8, -64.1]**
Finance & Accounting	Single-loop variant	*+300.0% [+0.0, +1100]*	—	-25.0% [-51.9, +9.5]
Finance & Accounting	Deep Sourcing (iterative)	***+877.5% [+268.3, +2110]***	—	**-57.5% [-72.2, -37.3]**
Other	Single-loop variant	+33.1% [-16.2, +117.7]	+818.4% [+104.1, +1328.6]	-13.7% [-31.5, +8.4]
Other	Deep Sourcing (iterative)	+64.7% [+9.4, +159.7]	+1263.6% [+222.0, +1793.9]	-9.1% [-25.3, +11.2]

categories.[4] These results are consistent with the hypothesis that the largest improvements are found in specialized roles where single-query retrieval is most limited.

The Finance & Accounting occupation shows some irregularities in Table 3 that are explained by small sample size. First, the confidence intervals for Prec(good)@10 are extremely wide for both the single-loop variant and Deep Sourcing. Second, in the baseline system, Prec(optimal)@10 was 0, so the relative lift in Prec(optimal)@10 for the test variants is undefined.

The Protective & Security occupation shows a lower bound of -100% for the Prec(optimal)@10 lift on the single-loop variant. This is also an artifact of small sample size; there are very few optimal matches in the baseline results for this category.

4.3. Downstream Outcomes

To evaluate Deep Sourcing against business metrics, we examined product workflows where employers can discover and contact candidates. These include search-based workflows where recruiters explicitly query pools of candidate profiles as well as matching workflows that proactively recommend profiles to recruiters. Data was collected over 12 weeks from a population of users who used Deep Sourcing at least once during that period.[5] The study tracked downstream outcomes for ~140,000 employer-initiated conversations, each encouraging a candidate to engage with a job. Three metrics were measured during the study: **response rate** is the fraction of contacted candidates who respond to the employer; **positive response rate** is the fraction of contacted candidates who respond expressing interest in the job; **apply rate** is the fraction of contacted candidates who submit a completed application for the job.

Results are shown in Table 4. Deep Sourcing is associated with substantial improvements in response rate and positive response rate across the evaluated product workflows, with higher rates than the search-based and matching baselines on these metrics. Deep Sourcing is also associated with a sizable lift in apply rate when compared with the search-based workflow. However, its apply rate is indistinguishable at the 95% level from that of the matching workflow, even though Deep Sourcing is associated with strong lifts on the two earlier-funnel rates against the same baseline.

One hypothesis for the null result comparing Deep Sourcing with the matching workflow is selection bias regarding which jobs are sourced by which workflow. To partially address that, we restrict our comparison so that each baseline workflow is evaluated only on the jobs that were the subject of employer-initiated conversations from both it and Deep Sourcing during the 12-week study window. This trades the statistical power of the unrestricted comparisons for tighter causal interpretation. The restricted evaluations of Deep Sourcing against the search-based and matching workflows are based on 2,710 and 3,570 conversations, respectively.

Selection bias is not fully mitigated by this restriction. Other potential sources of bias include which jobs were processed using multiple workflows, the recruiter's choice of which candidates identified by each workflow to contact, and possible differences in the messages sent through each workflow.

Results of the comparisons are shown in Table 5. Against the matching workflow, Deep Sourcing is associated with a +18.6% [+7.9, +29.8] relative lift in response rate, +20.5% [+6.6, +35.5] in positive response rate, and **+30.6% [+4.2, +61.4] in apply rate**. We interpret this as evidence that the unrestricted apply-rate null is substantially driven by

[4] excluding Finance & Accounting Prec(optimal)@10 where lift is undefined

[5] This is an observational study; recruiters chose which workflow to engage with for each sourcing task rather than being randomly assigned to Deep Sourcing.

Table 4
Downstream outcome comparison. Values are shown as relative lift of Deep Sourcing over the baseline workflow with 95% confidence intervals from a parametric binomial bootstrap.

Baseline workflow	Response rate	Positive response rate	Apply rate
Search-based	+23.1% [+17.8, +28.3]	+37.5% [+30.2, +45.0]	+117.8% [+94.8, +141.7]
Matching	+15.3% [+10.3, +20.4]	+13.4% [+7.0, +19.5]	*-1.4% [-11.9, +9.3]*
All (excluding Deep Sourcing)	+22.4% [+17.2, +27.5]	+30.5% [+23.6, +37.6]	+57.7% [+41.7, +74.8]

Table 5
Within-job downstream outcome comparison. Each row restricts to jobs that were the subject of employer-initiated conversations from both Deep Sourcing and the indicated baseline workflow during the 12-week study window: 2,710 conversations for the search-based comparison and 3,570 conversations for the matching comparison. Values are shown as relative lift of Deep Sourcing over the baseline workflow with 95% confidence intervals from a parametric binomial bootstrap.

Baseline workflow	Response rate	Positive response rate	Apply rate
Search-based	+6.3% [-3.9, +17.0]	+11.7% [-2.1, +26.8]	+27.9% [+2.1, +57.4]
Matching	+18.6% [+7.9, +29.8]	+20.5% [+6.6, +35.5]	**+30.6% [+4.2, +61.4]**

job-level selection: the matching workflow appears to have been used disproportionately for tasks with naturally higher apply rates. The within-job comparison against the search-based workflow shows the complementary pattern. Earlier-funnel lifts shrink (+6.3% [-3.9, +17.0] in response rate, +11.7% [-2.1, +26.8] in positive response rate with both CIs crossing zero) while the apply-rate lift remains substantially positive (+27.9% [+2.1, +57.4]). A plausible interpretation of this is that the unrestricted lifts against the search-based workflow are inflated by recruiters routing roles with naturally lower apply rates to that workflow.

5. Discussion

5.1. Feature Matrix Scoring

Early versions of Deep Sourcing asked the LLM to directly produce numeric match scores. This led to high variance across calls and inconsistent interpretation of the same candidate profile in different contexts, results that are consistent with the concerns identified in [30] over the validity and reliability of LLM-as-judge scoring. The feature matrix approach (Section 3.5) resolved this by having the LLM evaluate each criterion independently and computing the profile-match score deterministically. In a controlled comparison with $N = 177$, feature matrix scoring reduced RMSE of same-match scores by 27% compared to direct LLM scoring. This separation of semantic evaluation from deterministic aggregation is, in our experience, essential for production-quality LLM-based ranking.

5.2. Reflection Design

The reflection node's effectiveness depends on result transparency. Providing per-profile feature matrices and strategy-level context enables nuanced learning, but the size of that context is a trade-off. Larger contexts give the reflection node more to learn from, at the cost of latency and compute. Smaller contexts are cheaper but limit its ability to learn why strategies failed or whether deeper paging is likely to help. We currently place the top-K candidate profiles from each strategy (Section 3.7) in the reflection node's context, from which it selects the top-K overall. Because the strategies share scoring criteria, every profile that would be in the top-K by raw score is guaranteed to be in this context. The reflection node is not a merge-sort, however; its listwise reasoning ranks profiles by comparison rather than by raw score, producing a top-K that may differ from the score-ordered top-K.

5.3. Fairness

We implement several guardrails before deployment, during live operation, and after deployment, such as: job seeker location data is not sent in LLM context to prevent geographic bias; protected characteristics are not passed into the comparative reasoning node and its LLM instructions prohibit trying to infer them; evaluation criteria are derived from job requirements; a variety of automated responsible AI (RAI) guardrails are used to moderate employer-specified criteria, mitigating attempts to create problematic ones; all LLM inputs and outputs pass through our moderation layers; and reviews are conducted by our legal and RAI teams prior to launch with ongoing monitoring.

5.4. Limitations

Deep Sourcing's effectiveness depends on the underlying retrieval platform. If first-pass retrieval fails to surface relevant candidate profiles, LLM evaluation and reflection cannot recover. The system's reliance on multiple sequential LLM calls creates latency unsuitable for some real-time interactions, motivating the single-loop variant. LLM-as-judge evaluation, while more consistent than keyword matching, is not immune to documented biases [10, 29, 31] and requires ongoing monitoring.

The system also shares a broader risk faced by LLM-powered workflows: candidate-provided text may contain prompt-injection attempts or other adversarial content. Recent work has explored detecting [32] and mitigating [33] such attacks against a candidate ranking system.

5.5. Future Work

We are investigating replacing pointwise evaluation and reflection nodes with smaller, fine-tuned language models to reduce cost and latency. Early results suggest fine-tuned models can approach large LLM quality for structured evaluation tasks with well-defined rubrics. We are also exploring cross-session memory to accumulate employer preferences

and market knowledge, and ambient operation where the system proactively re-runs sourcing when hiring-pipeline health changes.

Acknowledgments

The authors would like to thank our colleagues who contributed to the development of Deep Sourcing, including Dan Arnett, Ken Blizzard-Caron, Halsey Lea, and Shannon Ling, as well as the many partners across Indeed who supported this work.

Declaration on Generative AI

During the preparation of this work, the author(s) used Google Gemini and Anthropic Claude Opus in order to: improve writing style, paraphrase and reword, manage citations, and generate simulated peer reviews. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, Okapi at TREC-3, in: D. K. Harman (Ed.), Proceedings of the Third Text REtrieval Conference (TREC-3), NIST Special Publication 500-226, National Institute of Standards and Technology, 1995, pp. 109–126. URL: https://trec.nist.gov/pubs/trec3/t3_proceedings.html.

[2] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Foundations and Trends in Information Retrieval 3 (2009) 333–389. URL: https://www.staff.city.ac.uk/~sb317/papers/foundations_bm25_review.pdf. doi:10.1561/1500000019.

[3] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2020, pp. 6769–6781. URL: https://arxiv.org/abs/2004.04906. doi:10.18653/v1/2020.emnlp-main.550. arXiv:2004.04906.

[4] J. Lin, R. Nogueira, A. Yates, Pretrained Transformers for Text Ranking: BERT and Beyond, Synthesis Lectures on Human Language Technologies, Springer, 2021. URL: https://arxiv.org/abs/2010.06467. doi:10.1007/978-3-031-02181-7, free preprint at https://arxiv.org/abs/2010.06467.

[5] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, K. Narasimhan, Tree of thoughts: Deliberate problem solving with large language models, in: Advances in Neural Information Processing Systems (NeurIPS), volume 36, 2023. URL: https://arxiv.org/abs/2305.10601. arXiv:2305.10601.

[6] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, ReAct: Synergizing reasoning and acting in language models, in: The Eleventh International Conference on Learning Representations (ICLR), 2023. URL: https://arxiv.org/abs/2210.03629. arXiv:2210.03629.

[7] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language models can teach themselves to use tools, in: Advances in Neural Information Processing Systems (NeurIPS), volume 36, 2023. URL: https://arxiv.org/abs/2302.04761. arXiv:2302.04761.

[8] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, S. Yao, Reflexion: Language agents with verbal reinforcement learning, in: Advances in Neural Information Processing Systems (NeurIPS), volume 36, 2023. URL: https://arxiv.org/abs/2303.11366. arXiv:2303.11366.

[9] A. Asai, Z. Wu, Y. Wang, A. Sil, H. Hajishirzi, Self-RAG: Learning to retrieve, generate, and critique through self-reflection, in: The Twelfth International Conference on Learning Representations (ICLR), 2024. URL: https://arxiv.org/abs/2310.11511. arXiv:2310.11511.

[10] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, I. Stoica, Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena, in: Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, volume 36, 2023. URL: https://arxiv.org/abs/2306.05685. arXiv:2306.05685.

[11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems (NeurIPS), volume 33, 2020, pp. 9459–9474. URL: https://arxiv.org/abs/2005.11401. arXiv:2005.11401.

[12] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, 2024. URL: https://arxiv.org/abs/2312.10997. arXiv:2312.10997.

[13] S.-Q. Yan, J.-C. Gu, Y. Zhu, Z.-H. Ling, Corrective retrieval augmented generation, 2024. URL: https://arxiv.org/abs/2401.15884. arXiv:2401.15884.

[14] S. Jeong, J. Baek, S. Cho, S. J. Hwang, J. Park, Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity, in: K. Duh, H. Gomez, S. Bethard (Eds.), Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 7036–7050. URL: https://aclanthology.org/2024.naacl-long.389/. doi:10.18653/v1/2024.naacl-long.389.

[15] A. Singh, A. Ehtesham, S. Kumar, T. T. Khoei, A. V. Vasilakos, Agentic retrieval-augmented generation: A survey on agentic RAG, 2025. URL: https://arxiv.org/abs/2501.09136. arXiv:2501.09136.

[16] L. Wang, H. Chen, N. Yang, X. Huang, Z. Dou, F. Wei, Chain-of-retrieval augmented generation, in: D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, N. Chen (Eds.), Advances in Neural Information Processing Systems, volume 38, Main Conference, Curran Associates, Inc., 2025, pp. 59888–59915. URL: https://proceedings.neurips.cc/paper_files/paper/2025/file/566aad4fac1acf17fd1ae8c3aef75326-Paper-Conference.

pdf. doi:10.52202/085713-2003.

[17] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, L. Yan, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, M. Bendersky, Large language models are effective text rankers with pairwise ranking prompting, in: Findings of the Association for Computational Linguistics: NAACL 2024, 2024. URL: https://arxiv.org/abs/2306.17563. arXiv:2306.17563.

[18] X. Ma, X. Zhang, R. Pradeep, J. Lin, Zero-shot listwise document reranking with a large language model, 2023. URL: https://arxiv.org/abs/2305.02156. arXiv:2305.02156.

[19] R. G. Reddy, J. Doo, Y. Xu, M. A. Sultan, D. Swain, A. Sil, H. Ji, FIRST: Faster improved listwise reranking with single token decoding, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2024, pp. 8642–8652. URL: https://aclanthology.org/2024.emnlp-main.491/. doi:10.18653/v1/2024.emnlp-main.491. arXiv:2406.15657.

[20] S. Zhuang, H. Zhuang, B. Koopman, G. Zuccon, A setwise approach for effective and highly efficient zero-shot ranking with large language models, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 38–47. URL: https://doi.org/10.1145/3626772.3657813. doi:10.1145/3626772.3657813.

[21] S. Chen, B. Jiménez Gutiérrez, Y. Su, Attention in large language models yields efficient zero-shot rerankers, in: The Thirteenth International Conference on Learning Representations (ICLR), 2025. URL: https://arxiv.org/abs/2410.02642. arXiv:2410.02642.

[22] C. Jin, H. Peng, S. Zhao, Z. Wang, W. Xu, L. Han, J. Zhao, K. Zhong, S. Rajasekaran, D. N. Metaxas, APEER: Automatic prompt engineering enhances large language model reranking, in: Companion Proceedings of the ACM Web Conference 2025 (WWW '25), ACM, 2025, pp. 2494–2502. URL: https://arxiv.org/abs/2406.14449. arXiv:2406.14449.

[23] J. Gao, B. Chen, X. Zhao, W. Liu, X. Li, Y. Wang, W. Wang, H. Guo, R. Tang, Llm4rerank: Llm-based auto-reranking framework for recommendations, in: Proceedings of the ACM on Web Conference 2025, WWW '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 228–239. URL: https://doi.org/10.1145/3696410.3714922. doi:10.1145/3696410.3714922.

[24] W. Jouanneau, E. Jouffroy, M. Palyart, An efficient long-context ranking architecture with calibrated LLM distillation: Application to person–job fit, in: Proceedings of the 5th Workshop on Recommender Systems for Human Resources (RecSys-in-HR 2025) co-located with the 19th ACM Conference on Recommender Systems (RecSys 2025), volume 4046 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2025. URL: https://ceur-ws.org/Vol-4046/RecSysHR2025-paper_1.pdf.

[25] F. P.-W. Lo, J. Qiu, Z. Wang, H. Yu, Y. Chen, G. Zhang, B. Lo, AI hiring with LLMs: A context-aware and explainable multi-agent framework for resume screening, in: CVPR 2025 Workshop, 2025. URL: https://arxiv.org/abs/2504.02870. arXiv:2504.02870.

[26] X. Yi, J. Yang, L. Hong, D. Z. Cheng, L. Heldt, A. Kumthekar, Z. Zhao, L. Wei, E. Chi, Sampling-bias-corrected neural modeling for large corpus item recommendations, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 269–277. URL: https://doi.org/10.1145/3298689.3346996. doi:10.1145/3298689.3346996.

[27] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), ACM, 2018, pp. 974–983. URL: https://arxiv.org/abs/1806.01973. doi:10.1145/3219819.3219890. arXiv:1806.01973.

[28] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37. URL: https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf. doi:10.1109/MC.2009.263.

[29] A. Panickssery, S. R. Bowman, S. Feng, LLM evaluators recognize and favor their own generations, in: Advances in Neural Information Processing Systems (NeurIPS), volume 37, 2024. URL: https://arxiv.org/abs/2404.13076. arXiv:2404.13076, oral presentation.

[30] K. Chehbouni, M. Haddou, J. C. Cheung, G. Farnadi, Neither valid nor reliable? investigating the use of llms as judges, in: D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, N. Chen (Eds.), Advances in Neural Information Processing Systems, volume 38, Main Conference, Curran Associates, Inc., 2025. URL: https://proceedings.neurips.cc/paper_files/paper/2025/file/829e8f32d76a6248815bf5b01633811d-Paper-Position_Paper_Track.pdf. doi:10.52202/085713-3029.

[31] P. Wang, L. Li, L. Chen, Z. Cai, D. Zhu, B. Lin, Y. Cao, L. Kong, Q. Liu, T. Liu, Z. Sui, Large language models are not fair evaluators, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 9440–9450. URL: https://aclanthology.org/2024.acl-long.511/. doi:10.18653/v1/2024.acl-long.511. arXiv:2305.17926.

[32] Y. Augey, J. H. Levy, A. Akdemir, RAPIDS: Resume attack prompt injection detection at scale, in: Y. Li, G. Rehm, M. Tu (Eds.), Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), Association for Computational Linguistics, San Diego, California, USA, 2026, pp. 1848–1865. URL: https://aclanthology.org/2026.acl-industry.127/. doi:10.18653/v1/2026.acl-industry.127.

[33] A. Akdemir, J. H. Levy, Understanding and defending against resume-based prompt injections in HR AI, in: Proceedings of the 5th Workshop on Recommender Systems for Human Resources (RecSys-in-HR 2025) co-located with the 19th ACM Conference on Recommender Systems (RecSys 2025), volume 4046 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2025. URL: https://ceur-ws.org/Vol-4046/RecSysHR2025-paper_9.pdf.