

Data Canvas: A Provenance-Guided Harness for Agentic Data Engineering

Zixuan Yi

University of Pennsylvania
zixy@cis.upenn.edu

Yuanming Shao

University of Pennsylvania
shaoym@cis.upenn.edu

Shaun Wallace

University of Rhode Island
shaun.wallace@uri.edu

Zachary G. Ives

University of Pennsylvania
zives@cis.upenn.edu

Ryan Marcus

University of Pennsylvania
rcmarcus@cis.upenn.edu

Abstract

As agentic LLM-based technologies mature, there is tremendous promise in using them to design and execute complex *data engineering* tasks, compiling and integrating specialized datasets on behalf of users. These tasks span from high-level planning with large reasoning models and query optimization, to lower-level runtime components for information extraction and semantic reasoning. Recent frontier systems show strong potential on such tasks but they remain prone to hallucinations, omissions, and underspecified decisions that are difficult to inspect or correct. We argue that agentic data engineering needs a provenance-guided harness: a control layer around planning and execution that makes outputs attributable, inspectable, and steerable through feedback. We present Data Canvas, a provenance-guided harness for agentic data engineering that wraps LLM-driven execution with structured semantic operators, explicit intermediate tables, and a provenance graph over operator events and results. This harness supports sparse human or automated feedback by tracing errors to their responsible reasoning steps, propagating corrections to related outputs, and replaying only the affected portions of the workflow. Using standard deep research benchmarks, we show that Data Canvas improves answer quality over existing systems, and that provenance-guided feedback yields substantial gains at a small fraction of the cost of prompt re-engineering and re-planning.

1 Introduction

Agentic LLM-based technologies promise to bring about a new wave of *agentic data engineering* tools (e.g., [Madden et al. \(2024\)](#); [Anderson et al. \(2024\)](#); [Shankar et al. \(2024\)](#)), analogous to agentic coding tools, but oriented around data synthesis tasks such as extracting structured data from unstructured sources, mapping heterogeneous data to a structured schema (e.g., [Arora et al. \(2023\)](#)), and cleaning and linking data. Such tasks, which have traditionally been addressed through extract-transform-load (ETL) tools or crowdworkers (e.g., [Wallace et al. \(2017\)](#); [Franklin et al. \(2011\)](#)), can now be tackled autonomously and at runtime.

Example 1

Suppose we want to help potential computer science graduate students identify potential PhD advisors. As a first step, we need to build a dataset of all computer science faculty who can advise students, across all R1 universities in North America, along with their research areas. For this task, a deep research system may work, but at the same time may: (1) provide incomplete answers, (2) hallucinate some answers, and (3) resolve ambiguity (e.g., the inclusion criteria for emeritus faculty) in unspecified ways, without consulting the user.

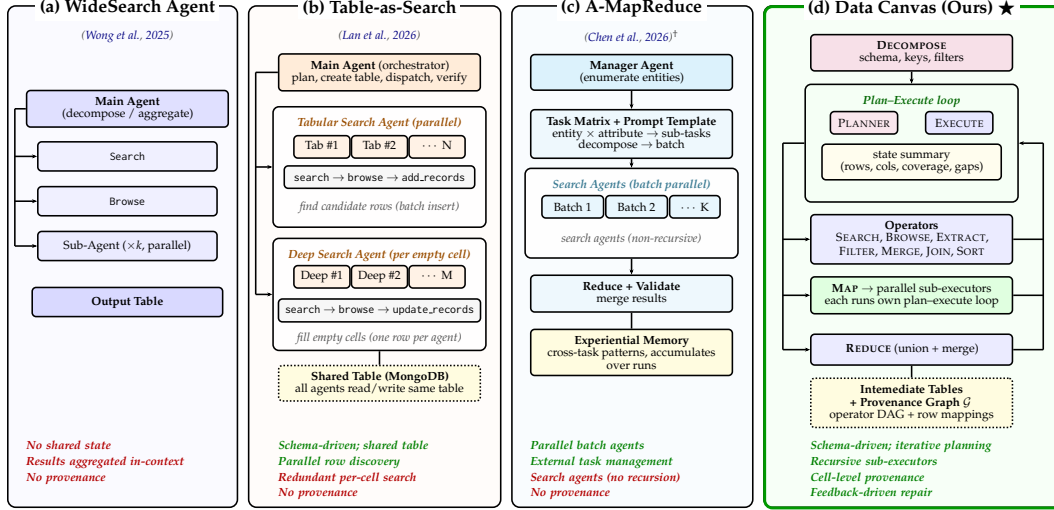
Compared to deep research systems (e.g., [OpenAI \(2025\)](#)), agentic data engineering approaches generally develop a more regularized, plan-based strategy for assembling and interpreting answers over large collections of sources — starting with high-level planning using techniques from large reasoning models, semantic query optimization ([Guo et al., 2025](#); [Russo et al., 2025](#)), and decomposition into table creation operators for “wide search” ([Wong et al., 2025](#); [Chen et al., 2026](#); [Lan et al., 2026](#)); then executing the plan using lower-level primitives for information extraction ([Shankar et al., 2024](#); [Anderson et al., 2024](#)), map-reduce ([Chen et al., 2026](#)), and semantic reasoning about collections of results ([Patel et al., 2024](#); [Trummer, 2025](#)). Importantly, expectations for soundness, completeness, and correctness are high — ultimately meaning that there is an unmet need for users to be able to **understand, assess, and repair model mistakes**. We argue that meeting this need requires a harness for agentic data engineering: a control layer around planning and execution that explains and attributes answers, accepts user feedback on output quality, and adapts reasoning traces dynamically. Inspired by the database provenance and view maintenance literature, we develop such solutions that combine user feedback on answers with information on how those answers were derived to efficiently repair errors.

- We propose Data Canvas, a provenance-guided harness that produces regular, structured LLM-driven plans with semantic operators to solve data engineering tasks.
- We develop strategies for tracking data provenance in a graph form, and for querying the derivation of results across various plan steps, back to their sources.
- We propose a method to incorporate sparse feedback and align model outputs by propagating credit or “blame” across the provenance graph to specific reasoning steps.
- Using standard deep research benchmarks, we show that our approach improves answer quality at a fraction of the inference cost of prompt re-engineering and full re-execution.

2 Prior Work

Large reasoning models, trained through reinforcement learning over human feedback ([Ouyang et al., 2022](#); [Ramesh et al., 2024](#)), have led to sophisticated “deep search” multistep reasoning systems ([Guo et al., 2025](#); [Yao et al., 2023](#)) that execute complex information-gathering tasks. These agents can be quite sophisticated, but their search strategies limit exploration, resulting in incomplete answer sets. In contrast, “wide search” ([Wong et al., 2025](#)) targets answer-assembly tasks with simpler, more regular plans but greater *breadth* and answer coverage: given a query and a target schema, an agent must exhaustively populate a results table from live web sources. DeepWideSearch ([Lan et al., 2025](#)) further combines breadth with multi-hop depth. Recent refinements of these ideas include TaS ([Lan et al., 2026](#)), which frames the task as table completion, WideSeek ([Huang et al., 2026](#)) and WideSeek-R1 ([Xu et al., 2026](#)), which train multi-agent policies via reinforcement learning, and A-MapReduce ([Chen et al., 2026](#)), which parallelizes query allocation with experiential memory. SODIUM ([Hu et al., 2026](#)) especially targets *depth within a domain* like a specific website: an agent constructs a result table by systematically navigating and extracting structured records, exploiting structural regularities across pages. MoNaCo ([Wolfson et al., 2026](#)) targets natural, complex questions that require reasoning across dozens of documents.

Data Canvas operates in the WideSearch setting — open-web, multi-source, schema-driven extraction. We create plans from *semantic operators* ([Patel et al., 2024](#); [Liu et al., 2025](#); [Anderson et al., 2024](#); [Russo et al., 2025](#); [Trummer, 2025](#)). Our work leverages formalisms for connecting operators to *provenance*. In the general case, an agentic plan resembles a



^{*}A-MapReduce is not directly evaluated; its experiential memory accumulates from prior tasks (all other evaluated systems are zero-shot).

Figure 1: Comparison of four broad information seeking paradigms. **(a)** WideSearch (Wong et al., 2025) is a multi-agent system. Results are aggregated in LLM context. **(b)** Table-as-Search (Lan et al., 2026) coordinates *Tabular Search Agents* and *Deep Search Agents* over a shared table. **(c)** A-MapReduce (Chen et al., 2026) enumerates entities in one shot and dispatches search agents in batch; an experiential memory accumulates across tasks. **(d)** Data Canvas (ours) decomposes the query, then runs a plan-execute loop. MAP launches recursive sub-executors, each with its own planning loop. A provenance graph $\mathcal{G}$ records operator DAGs and intermediate tables, enabling feedback-driven repair. None of (a–c) track provenance or support targeted error repair.

workflow pipeline, which can be captured in the PROV (Belhajjame et al., 2013) standard. The database literature developed a query-semantics-preserving treatment of provenance, where queries are limited to relational operators such as filters (selections), unions, and joins. “Why provenance” (Buneman et al., 2001) and provenance semirings (Green et al., 2007b) capture both *alternate derivations of an answer* and *derivation of a single answer from a combination of inputs*, and use this to support incremental updates (Green et al., 2007a). Our work adapts data provenance to the agentic setting, and uses it to trace incorrect outputs back to specific reasoning steps and sources, and performs *incremental view maintenance* (Lu et al., 1995) to propagate corrections. This is more targeted than global reinforcement learning, and more cost-effective than re-computation based on prompt updates (Khattab et al., 2024). PROV-AGENT (Souza et al., 2025) also studies provenance for agentic workflows, focusing on capturing and querying agent interactions, prompts, responses, and decisions. Data Canvas is complementary: it uses provenance for feedback-driven repair.

Recent research shows that adopting AI-assisted tools can positively motivate the contributions of paid crowdworkers (Wallace et al., 2025) to collect or improve the quality of tabular datasets (Papoutsaki et al., 2015; Wallace et al., 2021). Data Canvas’s provenance and feedback mechanisms can facilitate more collaborative interactions between crowdworkers and agentic data engineering pipelines.

3 Data Canvas

Data Canvas adopts a ReAct-based reasoning approach (Yao et al., 2023): Given a prompt, our system iteratively decomposes it into action sequences using a receding-horizon plan-execute loop. In our setting, each action is assembled from a small library of semantic operators (Table 1). Each semantic operator has access to the tabular results produced by previously executed semantic operators, and the operator produces its own tabular result. These intermediate relations are maintained as named artifacts in the plan state and materialized outside the LLM context as persisted table snapshots. Unlike prior semantic operators (Patel et al., 2024; Liu et al., 2025; Russo et al., 2025; Shankar et al., 2024; Dorbani

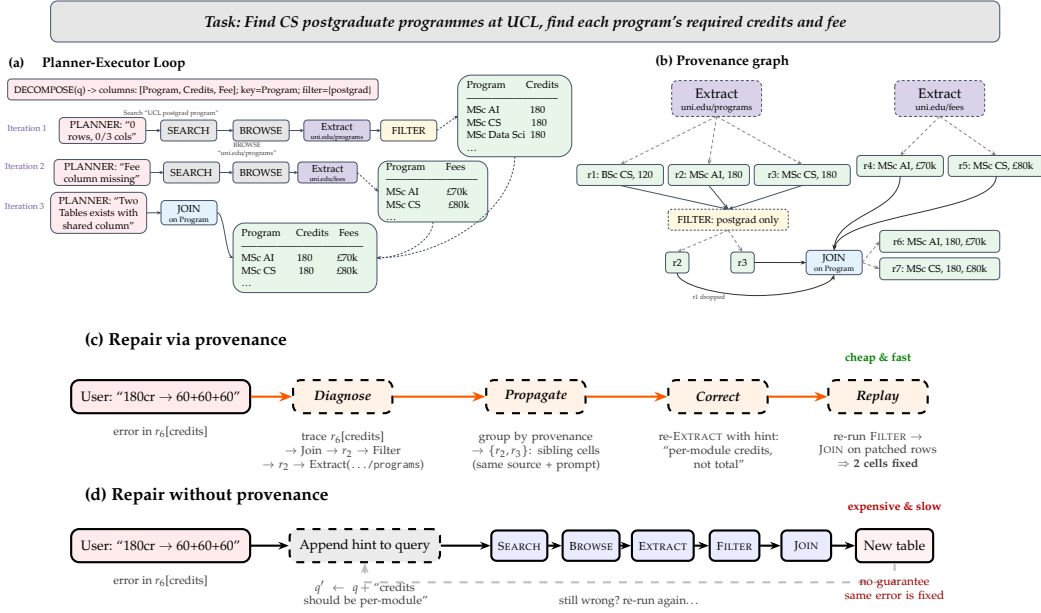


Figure 2: Provenance graph and repair. **(a)** The Planner-Executor loop builds a result table through operators and intermediate tables. **(b)** Provenance graph records how rows are derived from sources and operators. **(c)** Provenance enables targeted repair: a corrected cell can be traced, propagated, and replayed to fix related errors. **(d)** Without provenance, the only option is to append the correction to the query and re-run the entire pipeline.

et al., 2025), **each of our operators records provenance information:** any output can be algorithmically tied to operations and sets of inputs.

This distinction is important because feedback over LLM-generated tables is only useful if the system can determine where an output came from and which other outputs share the same cause. Many agentic systems expose execution only as a free-form trace or conversation log; asking an LLM to explain a value after the fact produces another probabilistic interpretation, which may itself be incomplete or wrong. Data Canvas instead records provenance during execution: rows and cells are linked by construction to the sources, intermediate artifacts, and operator invocations that produced them. This makes feedback operational rather than merely descriptive: a corrected output can be traced to its upstream cause, related outputs can be identified through shared provenance, and only the affected downstream computation needs to be replayed.

Example 2

Figure 2(a) shows how Data Canvas decomposes an agentic data engineering task ("Find CS postgraduate programmes at UCL; find each program's required credits and fee") into a series of semantic operators. In Iteration 1, tasks involve searching for pages, browsing to extract their text, and extraction of program credits. Iteration 2 defines a similar operator pipeline for fees. Iteration 3 joins the prior results together. Provenance is recorded during execution, resulting in the graph of Figure 2(b).

3.1 Iterative Planning and Execution in Data Canvas

Our main plan-execute loop is detailed in Algorithm 1. Given a query, the system first determines a target schema (columns, key columns, and filters). We reformulate the prompt with that target schema in mind, and alternate between planning and execution. The planner reasons over any existing *relational intermediate state* (artifact schemas, row counts, samples

Table 1: Semantic operators in Data Canvas.

$\text{SEARCH}(q) \rightarrow \mathcal{U}$	issues a web search query q ; returns URLs with titles and snippets.
$\text{BROWSE}(u) \rightarrow d$	fetches the content of URL u and returns the page text.
$\text{EXTRACT}(D, S) \rightarrow R$	given a set of documents D and a target schema S , calls an LLM to extract structured rows.
$\text{FILTER}(\sigma, R) \rightarrow R'$	applies a predicate σ to R , retaining matching rows. Predicates can be rule-based expressions or LLM-evaluated semantic conditions.
$\text{PROJECT}(R, \Pi) \rightarrow R'$	returns only the columns from R matching the set in Π .
$\text{MERGE}(R, K) \rightarrow R'$	deduplicates R on key columns K , using an LLM to resolve conflicts.
$\text{JOIN}(R_1, R_2, K) \rightarrow R'$	joins relations on shared key columns K , with optional LLM-based fuzzy matching.
UNION, SORT	standard relational operators.
$\text{MAP}(f, R) \rightarrow R'$	launches parallel sub-executors, per rows or columns, each running its own planning loop to enrich a subset of R .

Algorithm 1 DATACANVAS(q): Plan-Execute Loop

Require: Natural language query q
Ensure: Output relation R^*

```

1:  $\langle cols, K, \sigma \rangle \leftarrow \text{DECOMPOSE}(q)$  // schema, keys, filters
2:  $\mathcal{A} \leftarrow \emptyset$  // intermediate tables
3:  $H \leftarrow []$  // step history
4:  $\mathcal{G} \leftarrow \text{new ProvenanceGraph}$  // DAG
5: for  $i = 1, \dots, T_{\max}$  do
6:    $s \leftarrow \text{STATESUMMARY}(\mathcal{A}, H, q)$ 
7:    $\langle op, args, stop? \rangle \leftarrow \text{PLAN}(s)$  // LLM planner
8:   if  $stop?$  and  $\text{NOGAPS}(\mathcal{A}, cols)$  then
9:     break
10:  end if
11:  if  $op = \text{MAP}$  then
12:    Partition  $\mathcal{A}$  into  $\{P_1, \dots, P_n\}$  by row or column
13:     $\{R_j, \mathcal{G}_j\} \leftarrow \text{DATACANVAS}(P_j)$  in parallel // recursive
14:     $R \leftarrow \text{REDUCE}(R_1, \dots, R_n)$  // union the results
15:     $\mathcal{G}.\text{MERGESUBGRAPHS}(\mathcal{G}_1, \dots, \mathcal{G}_n)$ 
16:  else if  $op \in \{\text{SEARCH}, \text{BROWSE}, \text{EXTRACT}, \dots\}$  then
17:     $R \leftarrow \text{EXECUTE}(op, args, \mathcal{A})$ 
18:     $\mathcal{G}.\text{RECORDEVENT}(op, R)$ 
19:  end if
20:   $\mathcal{A}[args.out] \leftarrow R$ ; append step result to  $H$ 
21: end for
22:  $R^* \leftarrow \text{ENRICH}(\text{BEST}(\mathcal{A}), \mathcal{G})$  // fill remaining gaps
23: return  $R^*, \mathcal{G}$ 

```

of the entities, and per-column coverage from prior steps). It operates in a plan–execute–observe loop: at each iteration, it inspects the current tables (if any), selects one operator and its arguments, observes the result, and repeats. This enables targeted decisions: missing columns trigger MAP sub-agents for per-entity enrichment; shared key columns trigger a JOIN; diminishing returns trigger termination. After the planner completes, the executor dispatches the chosen operator, records provenance, and appends the result to the step history. The loop terminates when the planner proposes to stop with no coverage gaps remaining, or upon stagnation—consecutive iterations with no new rows or filled cells.

3.2 Provenance in Data Canvas

We represent provenance as a *directed acyclic graph* $\mathcal{G} = (V, A)$ of *data nodes* interleaved with *event nodes*, connected by *provenance arcs* (see Figure 2a for a concrete example). **Data nodes** d_i appear in lime color in the figure. These nodes' raw data is *not* actually stored in the graph itself. Instead, each data node has a *row ID* that references the on-disk snapshot of data items in the output relation of an operator.

Each **event node** e_j in V corresponds to a single operator invocation; an arc from $d_i \rightarrow e_j$ indicates that data row d_i was consumed as input by e_j . The operator may emit data nodes d_o ; each output d_o is connected via an arc $e_j \rightarrow d_o$. Certain operators annotate their event node with an *input-to-output row mapping* that links each input row ID to the output row IDs it produced (see the details below). Each event node $e \in V$ stores:

- An *operator type* (one of EXTRACT, FILTER, MERGE, JOIN, UNION).
- A *row mapping* $\mu_e : r_{\text{in}} \mapsto [r_{\text{out}}^1, \dots, r_{\text{out}}^k]$ linking each input row ID to the output row IDs derived from it. For selection-like operators, dropped rows map to $\perp$; for leaf events (EXTRACT), the input side is empty.
- An *output reference*: the CSV file containing the output relation, keyed by row ID.
- *Operator-specific metadata*: for EXTRACT, the source URL; for FILTER, per-row accept/reject decisions; for MERGE, group membership and per-column conflict resolution details.

Row IDs for output data nodes are UUIDs assigned at creation by EXTRACT. In fact, most operators preserve row IDs, with the exception of operators that combine input rows (MERGE, JOIN). The graph is constructed incrementally: each operator invocation adds an event node, arcs from its input events, and writes the output relation to a new CSV file (creating corresponding data nodes). Browsed web pages are retained in a browser cache keyed by URL to ensure reproducibility during repair.

Creating the Provenance Graph at Runtime. As query execution progresses, each operator updates the provenance graph. For example, SEARCH returns lists of sources, which can be consumed by BROWSE, emitting text strings, each of which is deterministically tied to a source (by construction). EXTRACT maps these strings to lists of tuples. For simplicity, we compact the browse-extract pair into a single event node for each source in Figure 2(a). For each subsequent operator, we typically create an event node:

- For tuple-to-tuple operators like FILTER, we create output data nodes for retained tuples and record a forward row mapping from each input row to its corresponding output row. If an input tuple is removed, we map it to $\perp$. This lets us represent selections uniformly while still identifying which rows were dropped.
- For operators that combine multiple input tuples (MERGE, JOIN), we create an output data node linked to the event node and record a forward row mapping from every contributing input row ID to the corresponding output row ID. Thus, multiple input rows may map to the same output row, naturally capturing many-to-one derivations such as deduplication and joins.
- SORT records input-to-output row ordering without adding value-level provenance.
- PROJECT preserves provenance for retained cells.
- MAP is represented by recursive sub-executor graphs that are merged into the parent graph after REDUCE.

Querying the Provenance Graph. The provenance graph documents execution for users and supports repairs based on user feedback on output results. To achieve this, the event-node DAG supports two key query patterns over forward provenance links.

Trace queries reconstruct the derivation path for a row or cell by walking the graph backward from event node to event node, while consulting the inverse of the forward row mapping at each step to recover the relevant predecessor row IDs. For example, tracing $r_6[\text{credits}]$ in Figure 2(a): r_6 appears in the output of JOIN; inverting that event's row mapping for r_6 yields the contributing inputs r_2 and r_4 . Since the credits column originates from the r_2 side,

we follow r_2 backward to FILTER, where inversion yields its predecessor row, and then to EXTRACT₁ (leaf), identifying the source URL `uni.edu/programs`.

Impact queries determine which rows are affected by a faulty event node. Given e^* identified by a trace, we use its forward row mapping to enumerate the output rows in the image of the implicated input rows; in the coarsest case, this is simply the full output relation of e^* . These are the *sibling rows* that were produced by the same operator invocation on the same source content, and are therefore likely to contain the same error.

3.3 Strategies for Provenance-Based Propagation of Feedback

Errors in LLM-based execution are *systematic*: the same source page, processed with the same prompt, produces the same class of mistake for every row with matching provenance. Our repair algorithm (Figure 2c) exploits this in three steps.

Diagnose. The provenance trace pinpoints which event node caused the error. We walk the graph backward from the error cell to the leaf BROWSE-EXTRACT, collecting the intermediate value at each event node. An LLM receives this trace, with the predicted value, gold value, and cached source page, and identifies the responsible node: e.g., a MERGE that resolved a conflict incorrectly, a FILTER that dropped a valid row, or a source that is irrelevant to the target population. The LLM produces a *correction hint* targeting the operator.

Propagate. One diagnosis identifies all rows sharing the same bug. The faulty event node’s forward row mapping enumerates its *sibling rows*—the outputs of the same operator invocation, or of the implicated inputs within that invocation—which likely exhibit the same systematic error.

Correct and Replay. The repair patches the faulty node and cascades the fix forward. For cell-level errors, we re-run the operator with the correction hint on the sibling rows loaded from the referenced CSV. For source-level exclusion, we remove the event node’s output rows entirely. Regardless, we traverse the graph forward, re-executing downstream event nodes in topological order. For missing entities, the system runs a targeted search-browse-extract pipeline. Section 4 shows reductions in cost of up to $35\times$ vs re-execution.

4 Experiments

We evaluate Data Canvas on two benchmarks, comparing against state-of-the-art agentic and deep research systems. Our experiments address three questions: (1) Does structured planning with provenance tracking improve table extraction quality over existing approaches? (2) Can sparse feedback repair, guided by lineage, improve quality at low cost? (3) How do cost, latency, and quality trade off across systems?

4.1 Experimental setup

We evaluate on two benchmarks. First, we use the English subset of the WideSearch benchmark (Wong et al., 2025), which contains diverse information-seeking queries that require extracting structured tables from the open web. Tasks span domains including education, technology, sports, travel, law, business, health, and science. Each query specifies a target schema (columns and expected rows), and gold-standard tables are provided for evaluation. Table sizes range from 11 to 533 rows and 3 to 14 columns. Second, we evaluate on the Drafty CS Professors dataset (Wallace et al., 2017)¹, a curated table of all computer science faculty. This task requires discovering faculty members from departmental web pages and extracting structured attributes (name, title, research area, etc.), providing a focused single-institution benchmark for entity-complete extraction.

Metrics. Following the WideSearch evaluation protocol, we report metrics at three granularities. *Row-level* recall, precision, and F1 measure whether complete rows are correctly retrieved, requiring all cells in a row to match the gold standard. *Item-level* recall, precision,

¹<https://drafty.cs.brown.edu/csprofessors>

Method	Row-level			Item-level			Overall		Latency (s)	Cost (\$)
	R	P	F1	R	P	F1	Ent. F1	Cell Acc.		
WideSearch	18.4	34.5	21.8	36.5	66.9	43.4	52.6	73.3	830.49	1.51
DeepResearch	10.7	14.7	11.7	18.1	26.3	20.2	23.3	29.2	268	0.83
TaS	22.7	35.7	25.6	44.2	66.3	49.6	60.4	73.5	3940	18.27
TaS(mini)	13.3	30.5	17.1	29.4	63.4	37.3	49.3	68.8	1835	1.93
Ours	24.5	32.8	26.6	49.0	66.7	52.8	63.2	74.0	2276	1.88
Ours+Repair	27.1	34.2	28.9	53.5	67.0	55.9	66.6	74.2	+234.64	+0.053

Table 2: Full evaluation on the WideSearch benchmark. All accuracy metrics are reported as percentages. Repair improves both precision and recall.

and F1 evaluate individual cell correctness across matched entity pairs. *Entity F1* measures whether the system retrieves the correct set of entities (rows), independent of attribute accuracy. *Cell accuracy* reports the fraction of individual cells that exactly match the gold values after normalization. All matching uses the WideSearch evaluation suite, which applies type-specific preprocessing (numeric extraction, date normalization, string canonicalization) and supports both exact match and llm judge.

Baselines. We compare against four systems:

- **WideSearch** (Wong et al., 2025): The reference system from the benchmark authors. We use the multi-agent framework with GPT-5 as the backbone model. This is the best-performing system on the benchmark.
- **DeepResearch** (OpenAI, 2025): OpenAI’s Deep Research, a deep research-style system that generates comprehensive answers but returns them in unstructured form, requiring post-hoc table extraction.
- **TaS** (Lan et al., 2026): Table-as-Search, a strong agentic baseline that reformulates information seeking as a table completion task, combining web search with iterative table construction. We re-ran their code using GPT-5 as the base model.
- **TaS (mini)**: A cost-reduced variant of TaS using GPT-5-mini as the backbone model, trading quality for lower inference cost.

Configuration. Data Canvas uses GPT-5 as the backbone LLM for planning and GPT-5-mini for extraction. Web search is performed via Google Search API, and interactive browsing uses Playwright. The planner operates with a receding horizon per iteration, replanning after each execution based on observed state. For repair experiments, we simulate sparse user feedback by sampling 1% of errors from the gold-standard table as corrections. Experiments were conducted on a machine with 32 CPU cores, without using the GPU.

4.2 Main results

Table 2 presents the full evaluation on the WideSearch benchmark, comparing Data Canvas against four baselines across row-level, item-level, and entity-level metrics, as well as latency and cost. Focusing on Data Canvas (without repair) versus the strongest baseline WideSearch, we observe that row recall improves from 18.4% to 24.5% and item recall from 36.5% to 49.0%, yielding a 4.8-point gain in row F1 (26.6 vs. 21.8) and a 9.4-point gain in item F1 (52.8 vs. 43.4) at comparable cost (\$1.88 vs. \$1.51). TaS achieves better row precision, but recall is low and at $9.7\times$ the cost (\$18.27) and $1.7\times$ the latency (3940s vs. 2276s), while DeepResearch scores only 11.7 row F1, confirming that unstructured research-style systems struggle with schema-conformant table extraction. *Recursive planning with operators improves both entity discovery and attribute completeness over existing approaches.*

Drafty CS Professors. Table 3 reports results on the Drafty CMU CS Professors task (Wallace et al., 2017), which requires discovering and extracting 7 attributes for all 240 computer science faculty members. We do not report TaS with GPT-5 on this task due to its prohibitive

Method	Row-level			Item-level			Overall		Latency (s)	Cost (\$)
	R	P	F1	R	P	F1	Ent. F1	Cell Acc.		
WideSearch	0.1	0.4	0.2	7.4	29.4	11.9	24.4	38.5	1823	4.75
DeepResearch	4.9	13.1	7.0	21.1	50.2	29.1	37.4	73.8	769	2.77
TaS (mini)	0.4	3.2	0.7	8.1	37.6	11.9	24.5	48.2	1588	2.33
Ours	24.0	21.5	22.7	62.7	55.9	59.1	74.4	75.1	5914	133.68
Ours+Repair	24.4	25.5	25.0	63.0	65.9	64.4	81.0	75.2	+5.2	+0.32

Table 3: Results on the Drafty CMU CS Professors task (240 gold rows, 7 columns). All accuracy metrics are reported as percentages. The repair here is excluding faculty only from computational biology. Data Canvas trace to the shared source and remove the FPs in one shot, substantially improving precision at very low cost.

cost.² Data Canvas achieves 75.1% entity F1 and 59.1% item F1, nearly tripling the entity F1 of both WideSearch (24.4) and TaS (mini) (24.5). DeepResearch attains 73.8% cell accuracy despite 27.4% entity recall; it returns few rows but gets them right. Our cost (\$133.68) of is higher than on WideSearch because the task requires per-entity enrichment: attributes like join year and doctoral institution are not on the department listing page and must be retrieved by browsing each professor’s individual profile (300 sub-tasks). This per-entity browsing cost is inherent to the task structure; the baselines that avoid it (DeepResearch, WideSearch) achieve much lower entity recall as a result. *Data Canvas discovers nearly 3× more entities than baseline systems, with cost scaling proportionally to the number of entities requiring individual page visits.*

4.3 Repair with sparse feedback

To evaluate the repair module, we simulate a human-in-the-loop setting. Given the system’s output table, a user inspects the output table and provides corrections for only a small fraction of the output error. In our experiments, we sample 1% of cell-level errors (identified by comparing against the gold table) as the feedback signal. The repair module then executes four steps: (1) *diagnose* each sampled error by tracing its provenance to the responsible operator and using an LLM to classify the root cause, (2) *propagate* by querying the lineage graph for all other cells sharing the same provenance path, (3) *correct* the faulty operator across all cells in the provenance group, and (4) *replay* the recorded pipeline from the corrected operator onward, re-running downstream operators with the patched intermediate results to produce the repaired table. The diagnosis step classifies each error into a root cause, leading to a repair strategy. We highlight three common categories:

- *Extraction ambiguity*: the source contains the correct information, but the extractor selects the wrong interpretation. This arises when the user’s query underspecifies how a column should be populated, while the gold table encodes a specific convention. For example, a query asks for “Credits” of university programs; the gold table expects per-module breakdowns (e.g., “60+60”), but the extractor returns the total (“120”). With one user-corrected cell, diagnosis identifies the mismatch and produces a hint (“extract individual credits and join with +”), which propagates to all rows extracted using the same prompt.
- *Filter recovery*: a filter operator incorrectly removes valid rows. When a missing entity is found among the filter’s dropped rows, this means the filter was too aggressive. The repair module restores the matched row directly, then uses the restored example as evidence to evaluate all other dropped rows from the same filter, recovering additional valid entities that were also incorrectly removed.

²On the WideSearch benchmark, TaS with GPT-5 costs \$18.27 per query for tables averaging only 20 rows. The 240-row Drafty task requires substantially more calls, making the cost impractical.

- *Merge conflict*: deduplication incorrectly resolves conflicting values. The repair re-resolves the conflict using the user correction as a tie-breaker, and propagates by re-evaluating other merge decisions within the same deduplication group.

Observations confirm that many errors in LLM-based operators are systematic. By repairing along a provenance path, a single feedback action can correct tens of results.

Results. The comparison between Ours and Ours+Repair measures the value of provenance-enabled feedback: both share the same initial execution, but Ours+Repair uses provenance to trace feedback to the responsible source/operator/row/cell, propagate it to related outputs, and replay only affected downstream steps. Table 2 (bottom two rows) shows that repair improves row F1 from 26.6 to 28.9, item F1 from 52.8 to 55.9, and entity F1 from 63.2 to 66.6, while adding only 234.64 seconds and \$0.053 per query. In contrast, incorporating feedback without provenance requires re-running the full pipeline with the correction appended to the prompt, costing \$1.88 and 2276 seconds per query—\$35 $\times$ more expensive and \$9.7 $\times$ slower than provenance-guided repair. *Sparse feedback repair guided by provenance achieves meaningful quality gains at 35x lower cost than re-execution.*

5 Discussion

Source discovery as the primary limitation. The most prevalent genuine failure mode is that web search does not surface the right sources, even when they exist. If the right pages are reached, the operator pipeline works well: cell accuracy on matched rows consistently falls between 80% and 99% across tasks. The bottleneck is upstream of the operators, bounded by search engine indexing and ranking. Improving source discovery (through query reformulation, source prioritization, or user-guided source selection) is a promising direction for future work. A portion of the remaining score gap reflects *temporal drift*: web pages update continuously while benchmark gold tables are frozen.

Provenance for interpretability and intent. Both source discovery failures and incorrect extractions typically share a common trait: the user’s intent is underspecified, and the system makes choices that may not match expectations. With provenance, the user can inspect *where* a value came from, judge whether the source or interpretation is acceptable, and issue a correction that propagates to all cells sharing the same provenance path.

Why not log-based repair? An alternative to provenance-guided repair is to provide the full execution log to an LLM and ask it to diagnose errors. Unfortunately, logs can span thousands of lines, making this approach costly. Worse, the LLM must *infer* which rows share the same error, a question that the provenance graph answers deterministically.

Feedback limitaion. Our sparse-feedback protocol idealizes which errors users notice by sampling from gold errors. The purpose was a controlled first evaluation of whether sparse feedback could be populated through provenance, not a full user study model.

6 Conclusion and Future Work

This work presented Data Canvas, an agentic data engineering system with human-guided repairs. Data Canvas can accurately extract large, tabular (relational) datasets from the open web in a cost-effective way, reducing costs by an order of magnitude over the current state of the art with improved accuracy. In future work, we plan to expand our data model to handle non-hierarchical joins, consider enhancements to data provenance, and investigate human-in-the-loop agentic data engineering systems.

Acknowledgments

This work is funded in part by NSF grants DBI-2400135 and IIS-2551799 and a gift from Google. Zixuan Yi is supported by an AWS ASSET Ph.D. Fellowship.

References

- Eric Anderson, Jonathan Fritz, Austin Lee, Bohou Li, Mark Lindblad, Henry Lindeman, Alex Meyer, Parth Parmar, Tanvi Ranade, Mehul A Shah, Ben Sowell, Dan G Tecuci, Vinayak Thapliyal, and Matt Welsh. The Design of an LLM-powered Unstructured Analytics System. *ArXiv*, abs/2409.00847, 2024. URL <https://api.semanticscholar.org/CorpusID:272368251>.
- Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *Proceedings of the VLDB Endowment*, 17(2):92–105, October 2023. ISSN 2150-8097. doi: 10.14778/3626292.3626294. URL <https://dl.acm.org/doi/10.14778/3626292.3626294>.
- Khalid Belhajjame, Reza B’Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. PROV-DM: The PROV Data Model, April 2013. URL <https://www.w3.org/TR/prov-dm/>.
- Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and Where: A Characterization of Data Provenance. In *ICDT*, pp. 316–330, 2001.
- Mingju Chen, Guibin Zhang, Heng Chang, Yuchen Guo, and Shiji Zhou. A-mapreduce: Executing wide search via agentic mapreduce, 2026. URL <https://arxiv.org/abs/2602.01331>.
- Anas Dorbani, Sunny Yasser, Jimmy Lin, and Amine Mhedhbi. Beyond Quacking: Deep Integration of Language Models and RAG into DuckDB, April 2025. URL <http://arxiv.org/abs/2504.01157>. arXiv:2504.01157 [cs].
- Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. CrowdDB: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD ’11, pp. 61–72, New York, NY, USA, June 2011. Association for Computing Machinery. ISBN 978-1-4503-0661-4. doi: 10.1145/1989323.1989331. URL <https://dl.acm.org/doi/10.1145/1989323.1989331>.
- Todd J. Green, Grigoris Karvounarakis, Zachary G. Ives, and Val Tannen. Update Exchange with Mappings and Provenance. In *VLDB*, 2007a.
- Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance Semirings. In *PODS*, 2007b.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojuan Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li,

- Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-025-09422-z. URL <https://www.nature.com/articles/s41586-025-09422-z>.
- Chuxuan Hu, Philip Li, Maxwell Yang, and Daniel Kang. Sodium: From open web data to queryable databases. *arXiv preprint arXiv:2603.18447*, 2026.
- Ziyang Huang, Haolin Ren, Xiaowei Yuan, Jiawei Wang, Zhongtao Jiang, Kun Xu, Shizhu He, Jun Zhao, and Kang Liu. Wideseek: Advancing wide research via multi-agent scaling. *arXiv preprint arXiv:2602.02636*, 2026.
- O. Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into State-of-the-Art Pipelines. In *International Conference on Learning Representations*, 2024. URL <https://api.semanticscholar.org/CorpusID:271532771>.
- Tian Lan, Bin Zhu, Qianghuai Jia, Junyang Ren, Haijun Li, Longyue Wang, Zhao Xu, Weihua Luo, and Kaifu Zhang. Deepwideseek: Benchmarking depth and width in agentic information seeking. *arXiv preprint arXiv:2510.20168*, 2025.
- Tian Lan, Felix Henry, Bin Zhu, Qianghuai Jia, Junyang Ren, Qihang Pu, Haijun Li, Longyue Wang, Zhao Xu, and Weihua Luo. Table-as-search: Formulate long-horizon agentic information seeking as table completion, 2026. URL <https://arxiv.org/abs/2602.06724>.
- Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, and Gerardo Vitagliano. Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing. CIDR ’25, Amsterdam, Netherlands, 2025. URL <https://www.vldb.org/cidrdb/papers/2025/p12-liu.pdf>.
- James J. Lu, Guido Moerkotte, Joachim Schue, and V. S. Subrahmanian. Efficient Maintenance of Materialized Mediated Views. In *SIGMOD*, pp. 340–351, 1995.
- Samuel Madden, Michael Cafarella, Michael Franklin, and Tim Kraska. Databases Unbound: Querying All of the World’s Bytes with AI. *PVLDB*, 17(12):4546–4554, 2024. doi: 10.14778/3685800.3685916.
- OpenAI. Deep research system card, February 2025. URL <https://cdn.openai.com/deep-research-system-card.pdf>.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, March 2022. URL <http://arxiv.org/abs/2203.02155>. arXiv:2203.02155.
- Alexandra Papoutsaki, Hua Guo, Danae Metaxa-Kakavouli, Connor Gramazio, Jeff Rasley, Wenting Xie, Guan Wang, and Jeff Huang. Crowdsourcing from Scratch: A Pragmatic Experiment in Data Collection by Novice Requesters. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 3:140–149, September 2015. ISSN 2769-1349. doi: 10.1609/hcomp.v3i1.13230. URL <https://ojs.aaai.org/index.php/HCOMP/article/view/13230>.

- Liana Patel, Siddharth Jha, Parth Asawa, Melissa Pan, Carlos Guestrin, and Matei Zaharia. Semantic Operators: A Declarative Model for Rich, AI-based Analytics Over Text Data. 2024. URL <https://api.semanticscholar.org/CorpusID:271218837>.
- Shyam Sundhar Ramesh, Yifan Hu, Iason Chaimalas, Viraj Mehta, Pier Giuseppe Sessa, Haitham Bou Ammar, and Ilija Bogunovic. Group Robust Preference Optimization in Reward-free RLHF, May 2024. URL <http://arxiv.org/abs/2405.20304>. arXiv:2405.20304.
- Matthew Russo, Sivaprasad Sudhir, Gerardo Vitagliano, Chunwei Liu, Tim Kraska, Samuel Madden, and Michael Cafarella. Abacus: A Cost-Based Optimizer for Semantic Operator Systems, May 2025. URL <http://arxiv.org/abs/2505.14661>. arXiv:2505.14661 [cs] version: 1.
- Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G Parameswaran, and Eugene Wu. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *arXiv preprint arXiv:2410.12189*, 2024.
- Renan Souza, Amal Gueroudji, Stephen DeWitt, Daniel Rosendo, Tirthankar Ghosal, Robert Ross, Prasanna Balaprakash, and Rafael Ferreira Da Silva. Prov-agent: Unified provenance for tracking ai agent interactions in agentic workflows. In *2025 IEEE International Conference on eScience (eScience)*, pp. 467–473. IEEE, 2025.
- Immanuel Trummer. Implementing Semantic Join Operators Efficiently, October 2025. URL <http://arxiv.org/abs/2510.08489>. arXiv:2510.08489.
- Shaun Wallace, Lucy Van Kleunen, Marianne Aubin-Le Quere, Abraham Peterkin, Yirui Huang, and Jeff Huang. Drafty: Enlisting users to be editors who maintain structured data. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 5(1): 187–196, 2017. doi: 10.1609/hcomp.v5i1.13300. URL <https://ojs.aaai.org/index.php/HCOMP/article/view/13300>.
- Shaun Wallace, Alexandra Papoutsaki, Neilly H. Tan, Hua Guo, and Jeff Huang. Case Studies on the Motivation and Performance of Contributors Who Verify and Maintain In-Flux Tabular Datasets. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2): 448:1–448:25, October 2021. doi: 10.1145/3479592. URL <https://dl.acm.org/doi/10.1145/3479592>.
- Shaun Wallace, Talie Massachi, Jiaqi Su, Dave B Miller, and Jeff Huang. Towards Fair and Equitable Incentives to Motivate Paid and Unpaid Crowd Contributions. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI ’25, pp. 1–32, New York, NY, USA, April 2025. Association for Computing Machinery. ISBN 979-8-4007-1394-1. doi: 10.1145/3706598.3714195. URL <https://dl.acm.org/doi/10.1145/3706598.3714195>.
- Tomer Wolfson, Harsh Trivedi, Mor Geva, Yoav Goldberg, Dan Roth, Tushar Khot, Ashish Sabharwal, and Reut Tsarfaty. Monaco: More natural and complex questions for reasoning across dozens of documents. *Transactions of the Association for Computational Linguistics*, 14: 23–46, 2026.
- Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. WideSearch: Benchmarking Agentic Broad Info-Seeking, August 2025. URL <http://arxiv.org/abs/2508.07999>. arXiv:2508.07999 [cs.CL].
- Zelai Xu, Zhexiong Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. Widesee-r1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning, 2026. URL <https://arxiv.org/abs/2602.04634>.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *International Conference on Learning Representations*, abs/2210.03629, 2023.