RECIPE: COLLABORATIVE AGENTS LEARNING REUSABLE PROCEDURES FOR CLINICAL INFORMATION PROCESSING AND EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Preparing structured clinical data for healthcare agents requires extracting predefined fields from heterogeneous notes and tables. This process is hindered by conflicting evidence, costly source discovery, and limited reuse of extraction experience across patients. We introduce ReCIPE (**Re**usable **C**linical **I**nformation **P**rocessing and **E**xtraction), a multi-agent system that learns reusable extraction procedures represented in an evolving typed dependency graph, RouteGraph. Field-semantic retrieval guides a compiler agent in proposing executable routes, while a verifier agent checks their trial outputs against patient evidence. GraphIE, our extraction algorithm, executes these routes with evidence propagation and consistency checks, supporting jointly required inputs and alternative routes. Accepted procedures accumulate across training patients without accessing target reference values or retaining patient-specific answers. We also introduce MIMIC-IV-Graph, a new benchmark pairing patient-specific source values with explicit source and target field definitions, covering 43 targets across four types. Starting from learned graph snapshots with admission-local route discovery, ReCIPE outperforms eight baselines in overall extraction quality on unseen patients under a shared language model backbone. Compared with the direct usage of LLM, it improves the overall score from 0.213 to 0.967. Per admission, it reduces test-time token consumption by 99.7% and LLM calls by 96.9%. In a blinded review by five physicians, 396 of 400 reviewed output differences with an oncology workflow combining manual abstraction and simple regular expressions are judged correct only for ReCIPE. These results support evolving, evidence-grounded procedures for accurate and efficient clinical data preparation.

1 INTRODUCTION

Training domain-specific agents relies on high-quality domain data. In healthcare, preparing such data often requires substantial manual effort to extract and reconcile evidence scattered across clinical notes and structured tables (Kwon et al., 2024). Reducing this manual burden calls for automating *multi-source clinical field extraction*: given a patient's available notes and tables, together with source and target field definitions, the task is to produce a structured record containing values for the predefined target fields. The objective is to extract these values *accurately*, respecting field definitions and patient-specific evidence, and *efficiently*, limiting language model token consumption and calls. In practice, different sources may provide conflicting candidate values, requiring reconciliation using timing and clinical context. Meanwhile, identifying suitable sources for each target can involve many unproductive extraction attempts. These inefficiencies can recur across patients, as execution experience is difficult to consolidate into reusable extraction procedures, leading to repeated source discovery and planning and limiting opportunities for continued improvement.

Existing approaches address complementary parts of these bottlenecks. Graph-based retrieval helps locate relevant evidence (Gutiérrez et al., 2025), while query and pipeline systems organize extraction through executable plans, reusable code, and pipeline rewrites (Urban & Binnig, 2024; Arora et al., 2023; Shahbazi et al., 2026; Shankar et al., 2025; Wei et al., 2026). Successful code-based experiences can also be retained to inform subsequent tasks (Shi et al., 2024). However, a procedure's prior success does not guarantee its applicability to a new patient; reliable reuse requires

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

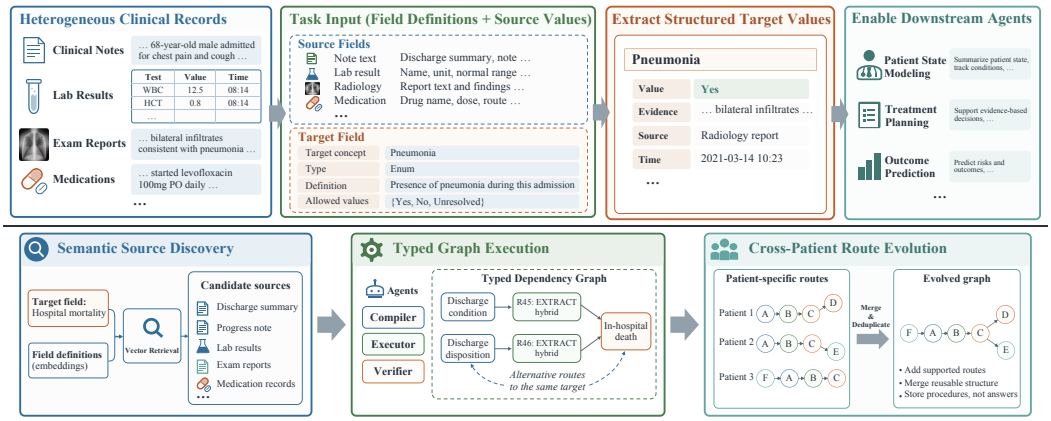


Figure 1: **Overview of ReCIPE. Top:** Given field definitions and heterogeneous records, the task is to extract structured target values with supporting evidence for downstream healthcare agents. **Bottom:** Field-semantic source discovery, typed graph execution, and route evolution accumulate verified procedures across training admissions; patient-specific values remain local. At test time, each admission starts from the same snapshot and permits local discovery without cross-admission retention.

explicit checks on source availability and output consistency with patient-specific evidence. Clinical verification and conflict-resolution methods address the need to check candidate values against patient evidence (Kwon et al., 2024; Gupta et al., 2025). Therefore, integrating source discovery, evidence-aware execution, and cross-patient improvement through reusable clinical field-extraction procedures remains largely unexplored.

To address these challenges, we introduce ReCIPE, a multi-agent system for discovering, executing, and improving clinical extraction procedures represented in a typed dependency graph, RouteGraph (Figure 1). To limit unproductive exploration during route discovery, semantic source retrieval first narrows candidate sources using vector similarity between source and target field definitions. Agents then use available operators to connect these sources, intermediate results, and targets in a typed directed acyclic graph (DAG). This graph distinguishes jointly required inputs from alternative routes, thereby specifying which inputs must be used together and which fallback routes can be attempted when expected sources are unavailable. Semantic retrieval thus guides source discovery, while typed dependencies specify how the retrieved sources can be used.

GraphIE executes the relevant subgraphs of RouteGraph while propagating source evidence alongside intermediate predictions, allowing candidate values to be checked against field definitions, types, timing, and clinical context. When cached procedures leave a field unresolved, GraphIE discovers additional routes using field-definition relevance to order candidate inputs. It reconciles conflicting candidates, attempts alternative routes when required sources are missing, and explicitly flags unresolved conflicts. During training, ReCIPE retains newly discovered procedures after checking their execution against the current patient's sources, without accessing hidden target values. RouteGraph stores these reusable procedures without retaining patient-specific answers. This reuse is intended to reduce repeated source discovery and planning, increasing efficiency.

Our setting requires a benchmark for extracting predefined clinical fields from mixed textual and tabular sources. However, benchmarks such as EHRSQL (Lee et al., 2022) focus on translating natural language requests into queries over structured records rather than evaluating this mixed-source extraction task. We therefore construct MIMIC-IV-Graph from the MIMIC-IV collection (Johnson et al., 2023c), covering four target field types: `Enum`, `Numeric`, `Datetime`, and `Text`. All methods receive the same field definitions and source-access permissions, and hidden reference values are reserved for evaluation. We compare our method with several baselines using the same language model backbone. Compared with Direct LLM, ReCIPE, using GraphIE to execute the learned RouteGraph, improves the overall extraction score from 0.213 to 0.967 (Table 1). Reported

average test-time token consumption per admission decreases by 99.7%, and LLM calls decrease by 96.9%.

Our contributions are threefold:

1. **A new benchmark for multi-source clinical field extraction.** We introduce MIMIC-IV-Graph for definition-driven extraction from mixed textual and tabular records, covering four target types and both direct and multi-step field recovery. It provides a common testbed for the upstream clinical data preparation needed by healthcare agents, under shared field definitions and source-access permissions.
2. **Typed dependency modeling with evidence-aware execution.** We propose a novel formulation coupling executable typed dependencies with source-evidence propagation and introduce GraphIE for conflict resolution and alternative-route execution grounded in field definitions and patient-specific evidence.
3. **Semantic source discovery and cross-patient route evolution.** To the best of our knowledge, we are the first to learn reusable clinical field-extraction procedures as evolving typed dependency graphs using execution feedback grounded in patient evidence. Semantic retrieval reduces unproductive exploration, while route evolution reduces repeated source discovery and planning across patients. ReCIPE achieves the best overall extraction performance among all evaluated methods on MIMIC-IV-Graph.

2 Related Work

Clinical Information Extraction. EHRSQL benchmarks queries over structured EHRs (Lee et al., 2022), while DrugEHRQA covers medication questions over notes and tables (Bardhan et al., 2022). CLEAR retrieves note excerpts using query-relevant entities (Lopez et al., 2025); EHRAgent generates and executes code for multi-table reasoning (Shi et al., 2024). CheckEHR checks note–table consistency (Kwon et al., 2024), and HARMON-E combines context-sensitive retrieval with hierarchical synthesis for oncology extraction across patient documents (Gupta et al., 2025).

Multi-Agent Systems. AutoGen supports tool-using agent conversations (Wu et al., 2024), MetaGPT coordinates specialized roles through standard operating procedures (Hong et al., 2024), and AgentVerse adapts expert teams (Chen et al., 2024). AFlow optimizes code-based workflows through Monte Carlo tree search and execution feedback (Zhang et al., 2025a). CAESURA compiles multimodal queries into execution plans (Urban & Binnig, 2024); OmniTQA combines relational and LLM-based semantic operators in cost-aware plans (Shahbazi et al., 2026). EVAPORATE-CODE+ ensembles synthesized extraction functions through weak supervision (Arora et al., 2023), DocETL rewrites document pipelines (Shankar et al., 2025), and MOAR jointly optimizes quality and cost (Wei et al., 2026).

Agent Memory. HippoRAG 2 combines passages and knowledge graphs for non-parametric factual and associative memory (Gutiérrez et al., 2025). Reflexion stores verbal feedback for later attempts (Shinn et al., 2023), ExpeL distills insights from task trajectories (Zhao et al., 2024), Voyager accumulates executable skills through environment feedback (Wang et al., 2024), and Agent Workflow Memory induces reusable routines for recurring subtasks (Wang et al., 2025). ReCIPE focuses on typed clinical procedures that reconcile source evidence, retain reusable routes, and revalidate outputs on new patients without storing patient-specific answers.

3 Methods

3.1 Task Formulation and the MIMIC-IV-Graph Benchmark

Task formulation. We study *multi-source clinical field extraction*: given source field values and definitions of both source and target fields, the task is to recover the target values. Let $\mathcal{S}$ and $\mathcal{T}$ denote fixed, disjoint sets of source and target fields, respectively. Their definitions $\mathcal{D} = \{d_f : f \in \mathcal{S} \cup \mathcal{T}\}$ specify field meaning, value type, cardinality, scope, and normalization rules. An episode e corresponds to a hospital admission, with source values $x_e = \{x_{e,s} : s \in \mathcal{S}\}$ comprising the permitted structured records and complete clinical notes, including discharge documentation for

retrospective extraction. Shared resources comprise a typed operator library $\mathcal{O}$ and reusable code-to-description dictionaries $\mathcal{K}$ (Appendix A). With reusable extraction state θ, a method produces

$$\widehat{Y}_e = f_\theta(x_e; \mathcal{D}, \mathcal{O}, \mathcal{K}) = \{(\hat{y}_{e,t}, \mathcal{E}_{e,t})\}_{t \in \mathcal{T}}, \tag{1}$$

where $\hat{y}_{e,t}$ satisfies the target's type and cardinality requirements, or is $\perp$ with an explicit failure status when unresolved; $\mathcal{E}_{e,t}$ identifies supporting source records or text spans. Intermediate targets contribute only their predictions and evidence. Reference values $y^\star_{e,t}$ are reserved for offline evaluation and unavailable during procedure construction, learning, or inference.

MIMIC-IV-Graph. MIMIC-IV-Graph combines textual and tabular sources from MIMIC-IV, MIMIC-IV-ED, and MIMIC-IV-Note. It contains 18 source fields and 43 targets: 11 `Enum`, 4 `Numeric`, 2 `Datetime`, and 26 `Text`. Patient-disjoint training and test sets contain 2,000 and 500 patients (2,487 and 620 admissions), retaining all eligible admissions per patient. All methods receive the same x_e, $\mathcal{D}$, $\mathcal{O}$, and $\mathcal{K}$. References use recorded table values, dictionary lookups, and fixed note-extraction rules; scoring measures agreement with these definitions and rules. Appendices C and E.3 detail cohort eligibility, dictionary provenance, field coverage, reference construction, and scoring exclusions.

Figure 2 summarizes ReCIPE: reusable procedures are represented in RouteGraph, executed and extended by GraphIE, and retained across training admissions through route evolution.

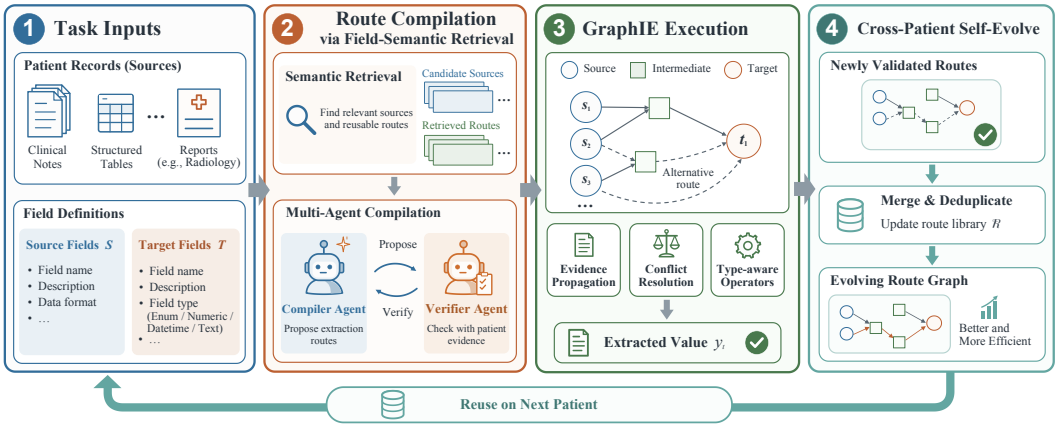


Figure 2: **ReCIPE workflow.** (1) Admission-specific records and source/target field definitions specify the extraction task. (2) Field-semantic retrieval guides the compiler's route proposals; the verifier checks trial outputs against patient evidence. (3) GraphIE executes RouteGraph with evidence propagation and conflict checks. (4) Accepted routes are merged and deduplicated for reuse across training admissions; patient-specific values remain local.

3.2 RouteGraph: Typed Dependency Graph Representation

Fields and executable routes. ReCIPE represents reusable extraction procedures in RouteGraph, a typed directed acyclic graph $G = (V, E)$ with field nodes and operator-instance nodes. The node set is $V = \mathcal{F} \cup \mathcal{R}$, where $\mathcal{F} = \mathcal{S} \cup \mathcal{T}$ contains the fields and $\mathcal{R}$ contains configured, reusable operator instances, which we call *routes*. Field nodes retain the types and cardinalities specified by $\mathcal{D}$. Each route $r \in \mathcal{R}$ specifies input fields I_r, one output target t_r, and an operator $o_r \in \mathcal{O}$. Its parameters ϕ_r configure the operator, for example through an extraction pattern, join predicate, or arithmetic formula:

$$r : \; I_r \xrightarrow{o_r, \phi_r} t_r, \qquad I_r \subseteq \mathcal{F}, \quad t_r \in \mathcal{T}. \tag{2}$$

Each route is represented by a single node, with incoming arcs from its input fields and an outgoing arc to its output target:

$$E = \bigcup_{r \in \mathcal{R}} \left(\{(f, r) : f \in I_r\} \cup \{(r, t_r)\} \right). \tag{3}$$

We call the edges in E *Field–Operator arcs*; a *Source-to-Target path* follows these arcs and may traverse multiple routes. Within a route, AND inputs express joint requirements and compatible OR inputs express alternatives under the operator contracts. Multiple routes to the same target provide alternative procedures. Intermediate target predictions can supply multiple downstream routes within admission e, while the graph stores reusable transformations rather than patient-specific values.

Operator semantics. The 11 fixed operators return candidate values, supporting evidence, and execution statuses. `EXTRACT` supports regex, LLM, and hybrid extraction; the other primitive operators are deterministic. `COMPOSITE` executes a bounded primitive sequence, possibly including model-based extraction, within one route node. Appendix A gives the operator contracts and a composite example.

3.3 GRAPHIE: PROVENANCE-AWARE INFORMATION EXTRACTION

GraphIE is ReCIPE's extraction algorithm: it executes routes in RouteGraph and discovers additional ones for unresolved targets. ReCIPE's compiler and verifier agents share this graph and use the same fixed language model under different prompts. They propose routes and check trial outputs against patient evidence, respectively. For admission e, GraphIE takes G, x_e, and targets $\mathcal{Q} \subseteq \mathcal{T}$, returning predictions with evidence, accepted additions Δ_e, and cost. GraphIE also supports an end-to-end wall-clock budget covering discovery, execution, and verification.

Evidence-aware execution. GraphIE first executes the requested targets' ancestor subgraph in dependency order, binding sources to x_e and intermediate inputs to current predictions. Each result carries supporting source records or text spans and must satisfy its field definition and evidence-reference checks. Successful routes resolve a target only when their normalized values agree; disagreement produces `CONFLICT`, while targets with no successful route retain an explicit execution status. These checks apply to both stored and newly proposed routes.

Field-semantic route discovery. For unresolved targets eligible for discovery, GraphIE narrows candidate inputs using cosine similarity between field-definition embeddings:

$$z_f = \frac{\mathrm{Enc}(d_f)}{\|\mathrm{Enc}(d_f)\|_2}, \qquad s(t,f) = z_t^\top z_f, \quad f \in \mathcal{F} \setminus \{t\}, \tag{4}$$

where d_f concatenates the field's name, description, type, cardinality, and unit. In our experiments, Enc is Qwen3-Embedding-4B (Zhang et al., 2025b). Search prioritizes available sources and resolved targets, and may introduce unresolved intermediates together with routes for computing them. The compiler agent jointly proposes routes for targets sharing candidate sources, using $\mathcal{D}$, operator and dictionary contracts, and graph metadata, but no patient values or clinical text. Discovery is bounded by round, proposal, and model budgets; similarity guides exploration rather than establishing source support.

Source-grounded validation and acceptance. Proposals first undergo static checks on input bindings, operator types, acyclicity, and input reachability. After trial execution, the verifier agent checks their outputs against complete relevant source records and applicable dictionary evidence. Dependency-connected additions are accepted together only if every route is supported and all outputs resolve. After rejected components are removed, previously resolved targets must remain resolved with unchanged values. Accepted routes update the admission-local graph and accumulate in Δ_e; these checks assess local consistency, not correctness on future patients. Algorithm 1 summarizes this loop, and Appendix B gives its state definitions, formal acceptance conditions, and detailed pseudocode.

3.4 CROSS-PATIENT ROUTE EVOLUTION

Routes accepted by GraphIE are stored in RouteGraph and form ReCIPE's reusable state θ in Eq. (1); patient-specific values and execution trajectories are excluded from the graph. Starting from $\mathcal{R}^{(0)} = \varnothing$, let $\mathcal{R}^{(n)}$ denote the routes stored after n admissions and Δ_{e_n} the accepted additions from the

Algorithm 1 GraphIE: execution and discovery overview

Require: G, x_e, targets $\mathcal{Q}$, resources $(\mathcal{D}, \mathcal{O}, \mathcal{K})$, discovery policy Λ, model budget B
Ensure: Predictions $P|_{\mathcal{Q}}$ with evidence and statuses, accepted routes Δ_e, cost c_e
1: $\widetilde{G}_e \leftarrow G$; $\Delta_e \leftarrow \varnothing$; $P \leftarrow$ execute the requested dependency subgraph on x_e
2: **while** eligible unresolved targets remain under Λ and search and model budgets permit **do**
3: Rank and expand inputs (Eq. (4)); compile routes from definitions and metadata
4: Statically validate proposals; trial-execute retained dependency components
5: Verify trial outputs against source records x_e and dictionary evidence $\mathcal{K}$
6: $A \leftarrow$ supported components passing the acceptance checks in Appendix B.4
7: Commit A and its arcs to $\widetilde{G}_e$; update P; $\Delta_e \leftarrow \Delta_e \cup A$
8: Advance search state, even when $A = \varnothing$
9: **end while**
10: **return** $(P|_{\mathcal{Q}}, \Delta_e, c_e)$, charging all model calls to B and c_e

next admission. When retention is enabled ($\rho = 1$),

$$\mathcal{R}^{(n+1)} = \mathrm{Unique}\Big(\mathcal{R}^{(n)} \cup \Delta_{e_n}\Big), \tag{5}$$

where Unique removes duplicate executable programs. Operator implementations and model weights remain fixed.

On each admission, GraphIE executes stored routes on that admission's source values and may discover additional routes for unresolved targets under Λ. Accepted additions update the local graph; with $\rho = 0$, they are discarded afterward and the shared graph remains unchanged. Appendix B details these controls and the update algorithm.

4 Experiments

4.1 Experimental Setup

Model and evaluation protocol. All LLM-based methods use locally deployed Qwen3.5-27B (Qwen Team, 2026). We evaluate admissions using the patient-disjoint 80%/20% split and shared Source records and field definitions described in Section 3.1. Target reference answers are reserved for evaluation. For Experiments 1–3, GraphIE starts each test admission from the same stored graph snapshot for the evaluated condition. Admission-local route discovery is enabled; all local additions are discarded afterward, with no retention across test admissions. Baseline adaptations and implementation settings appear in Appendices D and E.

Metrics and cost. We use macro-F1 for `Enum`, normalized exact match (EM) for `Numeric` and `Datetime`, and segmented BERTScore-F1 (Zhang et al., 2020) for `Text`. For Experiments 1–3, Overall is the equally weighted harmonic mean of the four type-level scores:

$$s_{\mathrm{overall}} = \frac{4}{s_{\mathrm{Enum}}^{-1} + s_{\mathrm{Numeric}}^{-1} + s_{\mathrm{Datetime}}^{-1} + s_{\mathrm{Text}}^{-1}}. \tag{6}$$

We report mean tokens and LLM calls per admission, including each method's verification and retries and, for ReCIPE, execution and admission-local discovery. Detailed scoring and cost rules appear in Appendix E.

4.2 ReCIPE Outperforms Baselines on MIMIC-IV-Graph

Table 1 compares ReCIPE with eight prompting, retrieval, workflow, and agent baselines under the shared model and source-access protocol in Section 4.1.

Higher extraction quality across all field types. ReCIPE ranks first across all four types, with Overall 0.967 versus 0.845 for CAESURA, the strongest baseline (+0.122). `Numeric` and `Datetime` EM reach 0.985 and 0.995, versus the best baseline scores of 0.721 and 0.867, gains of 0.264 and 0.128. `Enum` and `Text` improve by 0.037 and 0.016 over their respective best baselines. These gains cover exact-value recovery as well as text similarity.

Table 1: **Main results on the MIMIC-IV-Graph test set.** Metric definitions follow Section 4.1. Costs are averaged per test admission, with tokens reported in thousands. ReCIPE costs include execution, admission-local discovery, verification, and retries. **Bold** and underlined values indicate the best and second-best results in each column, respectively, including ties.

Category	Method	Extraction quality ↑					Cost ↓	
		Enum	Num.	Date	Text	Overall	Tokens (k)	Calls
Control	Direct LLM	0.235	0.102	0.419	0.433	0.213	917.26	83.276
Retrieval	Hybrid-RAG	0.333	0.262	0.714	0.510	0.393	349.34	91.053
	HippoRAG 2	0.337	0.262	0.704	0.521	0.395	420.01	95.748
Workflow	CAESURA	0.854	0.721	0.867	0.978	0.845	99.37	6.626
	EVAPORATE-CODE+	0.589	0.273	0.867	0.914	0.526	26.82	**1.981**
	OmniTQA	0.805	0.716	0.867	0.925	0.821	58.05	7.053
	DocETL	0.863	0.711	0.844	0.946	0.832	68.11	10.065
Agent	EHRAgent	0.827	0.716	0.867	0.976	0.836	131.77	9.550
Ours	**ReCIPE**	**0.900**	**0.985**	**0.995**	**0.994**	**0.967**	**2.84**	2.603

Substantially lower token consumption. ReCIPE averages approximately 2,840 tokens per admission, the lowest among all methods. This is 97.1% below CAESURA and 89.4% below EVAPORATE-CODE+, the baseline with the lowest token consumption. It uses 2.603 LLM calls per admission, the second-lowest among all methods and 60.7% fewer than CAESURA.

4.3 Evolved Extraction Routes Generalize to Unseen Patients

We evaluate RouteGraph snapshots G_p, $p \in \{0, 5, 10, 20, 30, 50, 80\}$, from a single training stream. Each snapshot is saved after

$$n_p = \left\lfloor N_{\text{train}} \frac{p}{80} \right\rfloor \tag{7}$$

training patients, with G_0 starting empty and G_{80} incorporating the full training partition. All snapshots use the same held-out patients, discovery policy, prompts, and model settings. Following Section 4.1, each admission permits local route discovery, with all local updates discarded afterward. A 40-second budget per admission covers search, execution, and verification.

Table 2: **Structural evolution of RouteGraph.** Source and Target field sets remain fixed. Routes count enabled, Source-reachable operator instances; arcs are Field–Operator arcs. Paths count Source-to-Target paths over all Targets, including intermediates. Counts describe the stored snapshot before admission-local discovery.

Snapshot	G_0	G_5	G_{10}	G_{20}	G_{30}	G_{50}	G_{80}
n_p	0	125	250	500	750	1250	2000
Routes	0	41	42	43	44	46	47
Arcs	0	96	98	108	110	114	116
Paths	0	55	56	65	66	68	69

Learned routes transfer to unseen patients. Figure 3 shows consistent improvements across `Enum`, `Numeric`, `Datetime`, and `Text`, with Overall increasing from 30.2 at G_5 to 96.7 at G_{80}, a gain of 66.5 points. Most of this gain emerges early: G_{20} reaches 89.3 after processing only one quarter of the training patients, followed by smaller, sustained improvements. Under the same discovery policy and time budget, snapshots containing more learned procedures achieve higher extraction quality on unseen patients without accumulating new routes across test admissions.

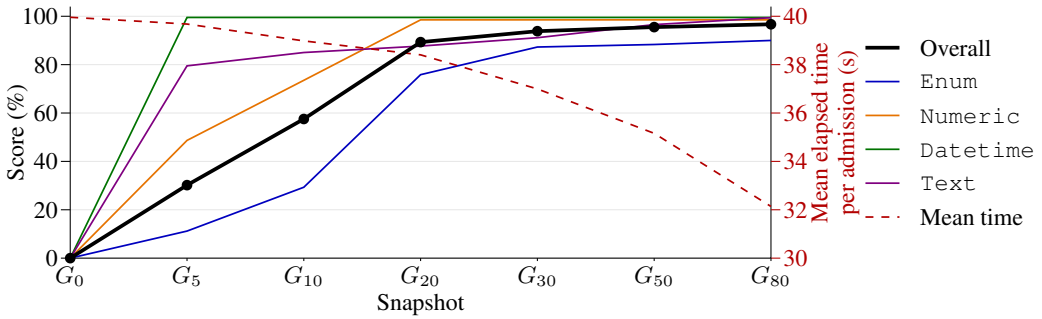


Figure 3: **Cross-patient transfer of evolved routes.** Each snapshot is evaluated on the same unseen patients with admission-local discovery and no cross-admission retention. The right axis reports mean elapsed time per admission (s) under a 40 s budget.

Table 3: **Learned versus physician-designed extraction programs on the MIMIC-IV-Graph test set.** Evaluation follows Section 4.1. Costs are per admission, with tokens in thousands; **bold** marks the best value in each column.

Graph	Extraction quality ↑					Cost ↓	
	Enum	Num.	Date	Text	Overall	Tokens (k)	Calls
G_{ref}	0.810	0.958	0.869	0.990	0.901	3.347	2.816
G_{80}	**0.900**	**0.985**	**0.995**	**0.994**	**0.967**	**2.840**	**2.603**

Evolution expands the repertoire of reusable routes. Table 2 shows that Source-to-Target paths increase from 55 at G_5 to 69 at G_{80}, alongside growth in Field–Operator arcs from 96 to 116. Over the same snapshots, the number of routes increases from 41 to 47. With the Source and Target fields unchanged, this growth enriches the procedures available for extracting existing fields rather than introducing new extraction tasks. Together, these results show how ReCIPE accumulates procedures from earlier patients in RouteGraph to improve extraction on unseen patients.

4.4 Learned Procedures Outperform a Physician-Designed Extraction Graph

Table 3 compares two independently constructed extraction programs: G_{80}, the RouteGraph learned by ReCIPE from training-set Source records, and the physician-designed reference G_{ref}. Two physicians jointly specify the reference DAG using the same field definitions and operator library, then manually review its executable operators and parameters. Both graphs are evaluated using GraphIE under the protocol in Section 4.1. Appendix F details their construction.

Higher extraction quality across field types. G_{80} achieves an overall score of 0.967, compared with 0.901 for G_{ref}, and scores higher on all four field types. Evaluating G_{80} uses approximately 2,840 tokens and 2.603 LLM calls per admission, outperforming G_{ref}. For ICU fields, both graphs use deterministic operators; a post-hoc diagnostic attributes their difference to coronary care unit (CCU) coverage in route parameters (Appendix F).

Case study: alternative evidence routes for mortality extraction. For hospital mortality (Figure 4), G_{ref} uses regex extraction from discharge condition, whereas G_{80} contains a hybrid route from that section (R45) and an additional route from discharge disposition (R46). Mortality Macro-F1 is 0.831 for G_{80} and 0.487 for G_{ref}. Paired outcomes identify 271 admissions correct only under G_{80} and two correct only under G_{ref}. In 264 of those 271 admissions, R46 is the recorded sole producing route for the mortality prediction. This route provides an additional, reusable way to obtain mortality evidence.

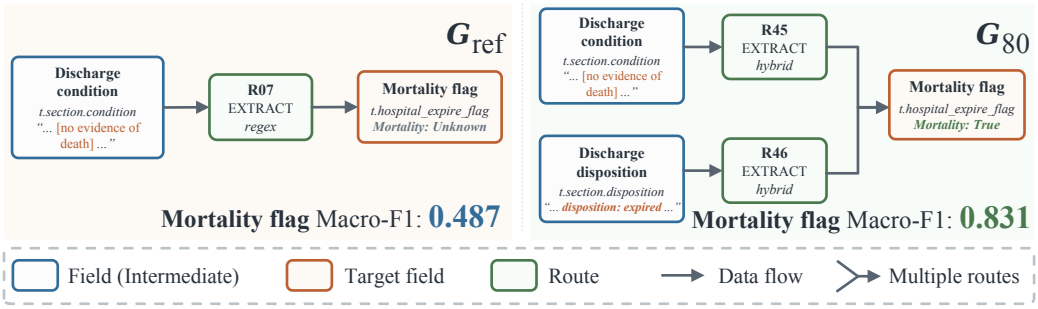


Figure 4: **Hospital-mortality extraction routes.** The physician reference uses discharge-condition evidence; G_{80} additionally accesses discharge-disposition evidence through R46. Only the relevant extraction subgraphs are shown.

Table 4: **Blinded physician review of differing outputs.** Counts reflect plurality votes from five physicians. *Both correct* denotes differing outputs that both satisfy the field definition and are supported by source evidence. Inter-rater agreement (%) averages, within each field type, the fraction of the 10 physician pairs choosing the same judgment per entry.

Field type	Sampled entries	Only ReCIPE correct	Only Reference correct	Both correct	Neither correct	Inter-rater agreement (%)
`Enum`	100	99	0	1	0	95.6
`Numeric`	100	100	0	0	0	99.2
`Datetime`	100	98	0	2	0	98.2
`Text`	100	99	0	1	0	96.4

4.5 Blinded Physician Review Favors ReCIPE on Differing Outputs

From a collaborating cancer center's oncology records, we randomly sample 100 pairs with differing ReCIPE and Reference outputs per field type (`Enum`, `Numeric`, `Datetime`, and `Text`). Reference combines manual abstraction with simple regular-expression extraction. Sampling precedes correctness assessment: five credentialed physicians independently judge each blinded pair against field definitions and source excerpts. Table 4 reports the four outcomes after plurality voting; Appendix G details the protocol.

Of 400 reviewed output differences, 396 (99.0%) are judged correct only for ReCIPE and four for both workflows. The advantage spans all four field types, with 98–100 outcomes per type favoring ReCIPE alone and none favoring Reference alone. The paired correctness difference is $\Delta_D = (396 - 0)/400 = 0.990$ (95% stratified bootstrap CI: $[0.980, 0.998]$). Mean pairwise inter-rater agreement ranges from 95.6% to 99.2%. In this deployment, ReCIPE extraction took one week, compared with approximately three months for Reference preparation.

5 Conclusion

ReCIPE learns reusable procedures in RouteGraph through field-semantic discovery, GraphIE execution, and source-grounded verification, without accessing target reference values or storing patient-specific answers. On MIMIC-IV-Graph, it outperforms eight baselines in Overall and test-time token consumption with admission-local discovery and no cross-admission retention, and its learned graph exceeds the physician-designed reference in extraction quality. In the oncology evaluation, physicians judge 396 of 400 reviewed output differences correct only for ReCIPE.

AI USE STATEMENT

We used generative AI tools to assist with literature search and retrieval, manuscript polishing, and experimental implementation and execution. This assistance did not extend to research ideation: the core research idea was conceived independently by the authors. To verify the AI-assisted work, we reviewed the resulting text and code and checked the reported experimental results against the saved evaluation artifacts. We take full responsibility for the final manuscript, including all text, code, results, and figures produced with AI assistance.

ETHICS STATEMENT

This study uses de-identified clinical data from MIMIC-IV, MIMIC-IV-ED, and MIMIC-IV-Note, obtained through credentialed access on PhysioNet. For the real-world evaluation, we additionally used retrospective oncology records from a collaborating cancer center, whose identity is withheld for double-blind review. The use of the cancer center's records for this study was reviewed and approved by its ethics committee, which also waived the requirement for informed consent. Access to and use of both the MIMIC datasets and the cancer center's records were subject to the applicable data use agreements and institutional requirements. The transmission of patient-level data to external services was strictly prohibited; all data processing, including model inference, was therefore performed locally.

REFERENCES

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. *Proc. VLDB Endow.*, 17(2):92–105, October 2023. ISSN 2150-8097. doi: 10.14778/3626292.3626294. URL https://doi.org/10.14778/3626292.3626294.

Jayetri Bardhan, Anthony Colas, Kirk Roberts, and Daisy Zhe Wang. DrugEHRQA: A question answering dataset on structured and unstructured electronic health records for medicine related queries. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 1083–1097, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.117/.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/578e65cdee35d00c708d4c64bce32971-Abstract-Conference.html.

Shashi Kant Gupta, Arijeet Pramanik, Jerrin John Thomas, Regina Schwind, Lauren Wiener, Avi Raju, Jeremy Kornbluth, Yanshan Wang, Zhaohui Su, and Hrituraj Singh. HARMON-E: Hierarchical agentic reasoning for multimodal oncology notes to extract structured data, 2025. URL https://arxiv.org/abs/2512.19864.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515. PMLR, 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Leo Anthony Celi, Roger Mark, and Steven Horng. MIMIC-IV-ED. *PhysioNet*, January 2023a. doi: 10.13026/5ntk-km72. URL https://doi.org/10.13026/5ntk-km72. Version 2.2.

Alistair Johnson, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV-Note: Deidentified free-text clinical notes. *PhysioNet*, January 2023b. doi: 10.13026/1n74-ne17. URL https://doi.org/10.13026/1n74-ne17. Version 2.2.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Brian Gow, Benjamin Moody, Steven Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV. *PhysioNet*, October 2024. doi: 10.13026/kpb9-mt58. URL https://doi.org/10.13026/kpb9-mt58. Version 3.1.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10:1, 2023c. doi: 10.1038/s41597-022-01899-x.

Yeonsu Kwon, Jiho Kim, Gyubok Lee, Seongsu Bae, Daeun Kyung, Wonchul Cha, Tom Pollard, Alistair Johnson, and Edward Choi. EHRCon: Dataset for checking consistency between unstructured notes and structured tables in electronic health records. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 89334–89345. Curran Associates, Inc., 2024. doi: 10.52202/079017-2835. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a291dad965af9bc2e39ffeb2ca1c6dc0-Paper-Datasets_and_Benchmarks_Track.pdf.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. EHRSQL: A practical text-to-SQL benchmark for electronic health records. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 15589–15601. Curran Associates, Inc., 2022. doi: 10.52202/068431-1134. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/643e347250cf9289e5a2a6c1ed5ee42e-Paper-Datasets_and_Benchmarks.pdf.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Ivan Lopez, Akshay Swaminathan, Karthik Vedula, Sanjana Narayanan, Fateme Nateghi Haredasht, Stephen P. Ma, April S. Liang, Steven Tate, Manoj Maddali, Robert Joseph Gallo, Nigam H. Shah, and Jonathan H. Chen. Clinical entity augmented retrieval for clinical information extraction. *npj Digital Medicine*, 8:45, 2025. doi: 10.1038/s41746-024-01377-1. URL https://www.nature.com/articles/s41746-024-01377-1.

Tom Pollard, Benjamin E. Moody, Li-wei H. Lehman, Brian J. Gow, Chrystinne Fernandes, Chen Xie, Alistair Johnson, Roger G. Mark, and Thomas Heldt. PhysioNet as a global platform for biomedical research. *Nature Health*, 1(8):792–795, August 2026. ISSN 3005-0693. doi: 10.1038/s44360-026-00096-z. URL https://doi.org/10.1038/s44360-026-00096-z.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Nima Shahbazi, Seiji Maekawa, Nikita Bhutani, and Estevam Hruschka. From textual columns to query plans: A unified relational-semantic execution framework for hybrid query processing, 2026. URL https://arxiv.org/abs/2604.02444.

Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. DocETL: Agentic query rewriting and evaluation for complex document processing. *Proc. VLDB Endow.*, 18(9):3035–3048, May 2025. ISSN 2150-8097. doi: 10.14778/3746405.3746426. URL https://doi.org/10.14778/3746405.3746426.

Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce C. Ho, Carl Yang, and May Dongmei Wang. EHRAgent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22315–22339, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1245. URL https://aclanthology.org/2024.emnlp-main.1245.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Matthias Urban and Carsten Binnig. CAESURA: Language models as multi-modal query planners. In *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, CA, USA, January 14-17, 2024*. www.cidrdb.org, 2024. URL https://www.cidrdb.org/cidr2024/papers/p14-urban.pdf.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Lindsey Linxi Wei, Shreya Shankar, Sepanta Zeighami, Yeounoh Chung, Fatma Ozcan, and Aditya G. Parameswaran. Multi-objective agentic rewrites for unstructured data processing. *Proc. VLDB Endow.*, 19(11):3062–3075, 2026. ISSN 2150-8097. doi: 10.14778/3836663.3836673. URL https://doi.org/10.14778/3836663.3836673.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025b. URL https://arxiv.org/abs/2506.05176.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

A Operator Library and Execution Contracts

Appendix organization. Appendices A and B specify the operator contracts and ReCIPE's algorithms. Appendix C describes benchmark construction and lists the fields. Appendices D and E describe the baseline adaptations, experimental protocol, and scoring rules. Appendices F and G provide the graph-comparison diagnostics and physician-review protocol.

A.1 Invocation Interface and Field Contracts

The fixed library $\mathcal{O}$ contains the 11 operator types in Table 5. A route binds named input ports to permitted Source fields or predicted Targets and configures an operator through parameters ϕ_r. For admission e, an invocation has the interface

$$(v_{e,r}, \mathcal{E}_{e,r}, \sigma_{e,r}) = o_r(X_{e,I_r}; \phi_r, d_{t_r}, \mathcal{K}), \tag{8}$$

where X_{e,I_r} contains the bound records, d_{t_r} is the output field definition, and $\mathcal{K}$ supplies the authorized dictionaries. The result consists of a candidate value $v_{e,r}$, evidence references $\mathcal{E}_{e,r}$, and a status $\sigma_{e,r}$. Each route declares one output Target, which may contain a sequence of records. GraphIE checks normalized outputs and source references, then propagates inherited evidence before publishing a successful Target prediction.

Types, identities, and scope. Port validation checks the input types required by each operator and the declared output type. Routes must use fields in the fixed vocabulary and preserve acyclicity; runtime checks enforce the Target's normalization and cardinality rules. Record-level operations align inputs by occurrence identity, preserving equal values at distinct occurrences. Cross-record joins declare the matching rule and the record that anchors the output. Input order alone does not establish temporal order. Units and clinical scope follow the field definitions and route parameters; arithmetic does not infer units or perform implicit unit conversion.

A.2 Operator Catalog

Table 5: Fixed operator library. Parameters select behavior within the declared implementations. Common status and input-availability rules are specified below.

Operator	Inputs and parameters	Output and conditions
`COPY_SELECT`	One input port; optional record index.	Copies all records or selects the specified position, retaining the input type and occurrence identities. Positional selection uses the supplied order.
`PARSE_NORMALIZE`	Value port; Target normalization; optional unit port, allowed unit aliases, and explicit value map.	Returns normalized values. Unparseable, unmapped, or unit-incompatible records are omitted. Numeric comparators are preserved; missing values receive no inferred category.
`LOOKUP`	Authorized dictionary and ordered key ports, such as code and version.	Aligns key components by record identity and returns the dictionary value with that identity. Absent keys produce no record. ICD lookup includes the code version.
`FILTER_JOIN`	Input fields; matching mode, predicates, selected port, scope, and cardinality policy.	Selects values using shared record keys, explicit value matches, or interval-duration matches. Cross-record outputs use the anchor identity. Ambiguities are skipped or rejected under the declared policy.

Continued on the next page

Table 5 (continued)

Operator	Inputs and parameters	Output and conditions
TIME_COMPUTE	Datetime start and end ports; elapsed_days formula.	Returns $(\text{end} - \text{start})/86400$ in days, with the difference measured in seconds. Inputs are aligned by record identity; timestamps must be parseable.
NUMERIC_COMPUTE	Two ordered Numeric ports; add, subtract, multiply, or divide.	Computes the declared formula on aligned records and normalizes the result. Operand order matters. Undefined arithmetic, including division by zero, fails execution.
AGGREGATE	Numeric records with sum, mean, min, or max; or value and time ports with previous_by_time.	Returns one aggregate value, or the preceding value for each later record. Temporal selection sorts by timestamp and then record identity; the first record has no predecessor.
EXTRACT	Text records; regex, LLM, or hybrid mode; patterns or instructions, scope, and occurrence-identity policy.	Extracts documented sections, entries, or attributes into the Target type. New occurrences use original-note spans; attributes can preserve their enclosing record identity. Model outputs require verifiable source quotations.
COMPOSE	Aligned input ports in a specified order; separator.	Concatenates existing values into Text, preserving record identity. It performs no additional clinical inference.
COALESCE_RESOLVE	Semantically equivalent alternative inputs with compatible types, units, cardinalities, and scope.	Ignores empty alternatives and returns the first available record collection when their value sets agree; otherwise returns CONFLICT. This comparison ignores record identities and duplicate multiplicity.
COMPOSITE	One to 12 primitive steps with input bindings, parameters, local result names, and optional local output contracts.	Returns the final step's output. Later steps can use earlier local results; execution stops at the first non-OK status. Nested composites are disallowed.

Composite example. A documented medication dose difference can be obtained using EXTRACT to recover doses, FILTER_JOIN to match medications and filter by unit, and NUMERIC_COMPUTE to subtract the admission dose from the discharge dose.

A.3 Input Availability and Execution Statuses

The AND input mode requires every mandatory input; the OR input mode requires at least one available input. The OR input mode is supported only for EXTRACT and COALESCE_RESOLVE, with matching input types, cardinalities, and units, plus declared semantic equivalence. If the route's input-availability condition or applicability guard fails, GraphIE returns NOT_APPLICABLE before invoking the operator. An invoked operator returns OK for a candidate, NOT_FOUND when no valid record remains, or CONFLICT for incompatible candidates. Multiple distinct values for a scalar Target also produce CONFLICT. INSUFFICIENT_EVIDENCE, EXECUTION_ERROR, and BUDGET_EXCEEDED distinguish unsupported output, failed execution, and exhausted limits.

A.4 Join Constraints, Model Use, and Composition

Join constraints. FILTER_JOIN supports equality, membership, nonempty-value, and regex predicates for records sharing an identity. Value-based joins can restrict matches to a note scope, normalize case and whitespace, and require unique anchor and candidate values. The interval-duration mode merges overlapping or contiguous eligible intervals and matches their elapsed days to a unique duration record at six decimal places. It omits ambiguous duration matches or contradictory boundary categories.

Model use and composition. Within the operator library, LLM calls occur only in `EXTRACT`, either directly or inside `COMPOSITE`. Hybrid extraction first runs regex matching and invokes the model only after `NOT_FOUND` or `INSUFFICIENT_EVIDENCE`; conflicts and execution errors do not trigger this fallback. Other primitive operations are deterministic given their inputs and parameters. Route discovery and semantic verification also call the model; these calls count toward the full procedure's cost. Composite-local values exist only during one invocation and do not expand the persistent field vocabulary. Reusable intermediate Targets are represented as graph nodes so that downstream routes can share their predictions and evidence.

B RECIPE ALGORITHMS AND IMPLEMENTATION

We specify the execution and update rules underlying Algorithm 1. The operator contracts appear in Appendix A.

B.1 ROUTE RECORDS AND ADMISSION STATE

The fields retain their declared types and cardinalities. In addition to I_r, o_r, and t_r from Equation 2, each route stores named input bindings, parameters ϕ_r, an input mode, an optional applicability guard, and a priority. Within a route, AND inputs express joint requirements and compatible OR inputs express alternatives under the contracts in Appendix A. Multiple routes to one Target provide alternative procedures. Reusable intermediate results occupy Target nodes, while variables internal to a composite operator remain local to that invocation.

For requested Targets $\mathcal{Q} \subseteq \mathcal{T}$, GraphIE maintains a local graph $\widetilde{G}_e$, a prediction map P, and an execution memo $\mathcal{M}_e$ for admission e. Each prediction contains a value, evidence, and status: $P_t = (\hat{y}_{e,t}, \mathcal{E}_{e,t}, \sigma_{e,t})$. Let $\mathcal{Z}(P)$ be the Targets with status `OK`, and let $\mathcal{S}_e$ be the Sources with nonempty records in x_e. These admission-specific objects are separate from the persistent graph G.

Structural counts and execution trajectories. A route definition is reused across admissions; a route invocation applies it to the current inputs. Structural counts treat all Source nodes as available and include enabled routes whose input requirements can be satisfied from Sources. Field–Operator arcs are the graph edges in Equation 3. A projected field-to-field dependency is an ordered pair (f, t_r) with $f \in I_r$; duplicate pairs across routes are merged. This projection omits operator nodes, parameters, and input-grouping semantics. Source-to-Target paths are counted over all Targets, including intermediate Targets. A path through an AND route covers only one input branch; executing that route requires all mandatory inputs. Structural Target coverage is the fraction of Targets reached by at least one such path. An execution trajectory (or execution trace) records the operations performed, observations, and statuses for one admission, including discovery and verification attempts when enabled. It can involve multiple routes and remains separate from the reusable route definitions.

B.2 DEFINITION-GUIDED ROUTE DISCOVERY

GraphIE first executes the cached routes in the ancestor subgraph of $\mathcal{Q}$ (Algorithm 3). For fields requiring discovery, it ranks candidate inputs using embeddings of each field's name, description, type, cardinality, and unit. With encoder Enc and definition text d_f, the ranking score is

$$z_f = \frac{\mathrm{Enc}(d_f)}{\|\mathrm{Enc}(d_f)\|_2}, \qquad s(t, f) = z_t^\top z_f, \quad f \in \mathcal{F} \setminus \{t\}. \tag{9}$$

Each field definition is serialized as its name, description, type, cardinality, and unit, separated by newline characters. Source and Target definitions use the same encoding procedure. We cache the embeddings for reuse across admissions and apply ℓ_2 normalization when building the index. Encoding inputs contain only public field metadata, without patient-specific values. Similarity orders exploration; source support is checked after execution.

Availability-aware expansion. Let R be the round limit and k_0 the initial candidate width. In round j, the base candidate width is $k_j = k_0 2^j$. Selection first takes up to $\max(1, \lfloor k_j/2 \rfloor)$ available Sources, then fills the remaining base slots from $\mathcal{S}_e \cup \mathcal{Z}(P)$ in similarity order. This reserves access

Algorithm 2 GraphIE with definition-guided route discovery

Require: Persistent graph G, admission Sources x_e, requested Targets $\mathcal{Q}$, policy Λ, model budget $\mathcal{B}$
Ensure: Predictions $P|_{\mathcal{Q}}$ and accepted new routes Δ_e; G remains unchanged

$\widetilde{G}_e \leftarrow G$; $\Delta_e \leftarrow \varnothing$; $\mathcal{M}_e \leftarrow \varnothing$; rejection history $H \leftarrow \varnothing$
$P \leftarrow \textsc{ExecuteGraph}(\widetilde{G}_e, x_e, \mathcal{Q}, \mathcal{M}_e)$
for $j = 0, \ldots, R-1$ **do**
  $U \leftarrow$ unresolved ancestor Targets eligible under Λ and current P
  if $U = \varnothing$ or $\mathcal{S}_e \cup \mathcal{Z}(P) = \varnothing$ **then**
    break
  end if
  $C_t \leftarrow \textsc{CandidateInputs}(t, j, \mathcal{S}_e, \mathcal{Z}(P), \Lambda)$ for each $t \in U$
  for each group B sharing candidate inputs **do**
    Refresh B and its candidates using the current prediction map
    Skip empty groups and unchanged contexts with no new program
    Stop discovery if a search limit or model budget is exhausted
    $\mathcal{P} \leftarrow \textsc{Compile}(B, \{C_t\}_{t \in B}, \mathrm{metadata}(\widetilde{G}_e, P, H))$
    Remove duplicate or malformed routes and routes to already resolved Targets
    $(G', P', A) \leftarrow \textsc{VerifyProposal}(\widetilde{G}_e, P, \mathcal{P}, x_e, \mathcal{Q}, \mathcal{M}_e)$
    $(\widetilde{G}_e, P) \leftarrow (G', P')$; $\Delta_e \leftarrow \Delta_e \cup A$
    Update H, duplicate-context records, and the proposal counters
  end for
end for
Attach stopping diagnostics and any model-budget or service-failure statuses
return $(P|_{\mathcal{Q}}, \Delta_e)$

to usable Sources even when unresolved Targets have higher similarity. For $j > 0$, search also adds up to $\max(1, \lfloor k_j/2 \rfloor)$ highly ranked unresolved intermediate Targets. Each receives up to two of its highest-ranked available Sources as candidate inputs. The compiler must supply procedures for any unresolved intermediate it uses. In our experiments, search uses one round and at most eight proposal batches per admission. Additional rounds widen the candidate neighborhood without enumerating the full program space.

Batched compilation and search control. Targets with overlapping Source candidates are grouped into bounded proposal batches. The compiler receives field definitions, operator and dictionary contracts, existing procedures, resolved-field identifiers, and rejection metadata. Patient values and clinical text enter execution and verification, not the compilation request. Each proposed route has one Target output and may contain jointly required inputs or a bounded composition of operators. Duplicate programs and routes whose outputs are already resolved are removed before evaluation. An unchanged context that yielded no new program is skipped on subsequent attempts.

Targets without enabled routes receive search priority; execution and evidence failures also remain eligible. For a Target with a cached route, `NOT_FOUND` or `NOT_APPLICABLE` can reflect admission-specific absence. A fixed hash of the case identifier selects approximately one in sixteen admissions to revisit these outcomes, with at most one such proposal batch per selected admission. This schedule uses no reference answers or cross-admission prediction state. The outputs remain unresolved, rather than verified clinical negatives. An admission may finish with missing outputs and no cache update.

Round and proposal limits stop discovery while preserving its current predictions, including normal absences. Model-budget exhaustion and service failures are recorded separately. Proposal, extraction, verification, and retry calls share admission-level accounting. Algorithm 2 combines these steps, with search policy Λ specifying candidate widths, grouping limits, and the cached-miss schedule. All subroutines share the public contracts and model budget $\mathcal{B}$; the compiler receives only metadata about dictionaries and execution.

B.3 Dependency-Ready Execution

Dependency-ready execution. ExecuteGraph collects the enabled routes needed by $\mathcal{Q}$ and its Target ancestors. A route becomes ready when every upstream Target has a settled prediction or

Algorithm 3 EXECUTEGRAPH: dependency-ready execution and evidence propagation

Require: Local graph G, Sources x_e, requested Targets $\mathcal{Q}$, admission memo $\mathcal{M}_e$
Ensure: Prediction map P for $\mathcal{Q}$ and its Target ancestors

$N \leftarrow$ Target ancestor closure of $\mathcal{Q}$; collect its enabled routes
Initialize P; settle Targets with no enabled route as `NOT_FOUND`
while some Target in N remains unsettled **do**
    Launch dependency-ready routes while fewer than W route workers are occupied
    Each invocation checks input availability and guards, then reuses or executes its operator
    Collect the next completed route observations
    for each completed route r with observation c_r **do**
        Memoize c_r unless execution failed or its model budget was exhausted
        if c_r has status `OK` **then**
            Check normalization and record/span references; mark failures explicitly
            If valid, attach the evidence inherited from upstream Target predictions
        end if
        Record c_r and decrement the pending-route count for t_r
    end for
    for each unsettled Target t whose candidate routes have all finished **do**
        $P_t \leftarrow$ RECONCILE$(\{c_r : t_r = t\}, d_t)$
        Release downstream routes waiting for t
    end for
end while
return P

failure status. Up to W ready routes execute concurrently; independent branches can proceed as soon as their dependencies settle. The AND input mode requires all mandatory inputs, while the OR input mode requires at least one available input. An unsatisfied input condition or applicability guard yields `NOT_APPLICABLE`. Each invocation reads only its bound Source records and upstream predictions. The memo $\mathcal{M}_e$ reuses route observations within the admission when both the route and its ancestor programs are unchanged.

Evidence and candidate reconciliation. For a successful operator output, GraphIE checks the Target's normalization and cardinality contract and validates every evidence reference. Structured references must identify existing records with matching values; text references must identify valid offsets whose substrings match the supplied quotations. Evidence inherited from intermediate Targets accompanies downstream predictions. A Target is finalized only after all its enabled candidate routes finish, including alternatives that complete later. If all successful candidates agree after normalization, GraphIE returns the first candidate in the fixed priority and route-identifier order. Distinct normalized outputs produce `CONFLICT`; priority cannot override disagreement. For sequence fields, this comparison preserves occurrence identities and duplicate multiplicity. If no route succeeds, GraphIE returns an explicit failure status as specified in Appendix A.

B.4 SOURCE-GROUNDED VERIFICATION AND ACCEPTANCE

Source-grounded acceptance. New procedures undergo additional checks before entering the local graph. Static validation enforces the field vocabulary, input ports, operator types, and acyclicity, and rejects outputs unrelated to the requested dependency closure. Structural reachability is checked from available Sources and resolved Targets before trial execution. The verifier then reads the proposed procedures, their published outputs, relevant intermediate predictions, and complete Source records in their dependency closure. Applicable lookup results supply dictionary evidence. For each route, it returns `SUPPORTED`, `REFUTED`, or `UNKNOWN`, checking meaning, occurrence coverage, timing, units, joins, and reusable parameters. The compiler and verifier are instructed to exclude patient-specific constants from reusable procedures. Cached procedures retain the mechanical execution checks above; this additional semantic assessment applies to newly proposed procedures.

New routes connected by dependencies form an atomic component, including connections through cached routes. Every new route in a component must receive `SUPPORTED`, and all of its output Targets must be resolved. Independent supported components can be retained when another component fails. After removing rejected components, GraphIE re-executes the retained graph if needed and

Algorithm 4 VERIFYPROPOSAL: component-wise source verification and acceptance

Require: Local graph G, predictions P, proposed routes $\mathcal{P}$, Sources x_e, Targets $\mathcal{Q}$, memo $\mathcal{M}_e$
Ensure: Updated local graph and predictions, and accepted routes A

Partition $\mathcal{P}$ into components linked by Target dependencies
$\mathcal{C} \leftarrow$ components passing static, relevance, and reachability checks
if $\mathcal{C} = \varnothing$ **then**
    return $(G, P, \varnothing)$
end if
$G^{\text{trial}} \leftarrow G \oplus \bigcup_{C \in \mathcal{C}} C$ ▷ $\oplus$ adds routes and incident arcs
$P^{\text{trial}} \leftarrow \text{EXECUTEGRAPH}(G^{\text{trial}}, x_e, \mathcal{Q}, \mathcal{M}_e)$
$\mathcal{C}^+ \leftarrow \{C \in \mathcal{C} :$ every output Target of C has status `OK`$\}$
if $\mathcal{C}^+ = \varnothing$ **then**
    return $(G, P, \varnothing)$
end if
$v \leftarrow \text{VERIFYAGAINSTSOURCES}(\mathcal{C}^+, G^{\text{trial}}, P^{\text{trial}}, x_e, \mathcal{D}, \mathcal{K})$
$A \leftarrow \bigcup\{C \in \mathcal{C}^+ : v_r =$ `SUPPORTED` for every $r \in C\}$
if $A = \varnothing$ **then**
    return $(G, P, \varnothing)$
end if
$G' \leftarrow G \oplus A$; $P' \leftarrow P^{\text{trial}}$
if some trial routes were rejected **then**
    $P' \leftarrow \text{EXECUTEGRAPH}(G', x_e, \mathcal{Q}, \mathcal{M}_e)$
end if
if $\text{Stable}(P, P')$ and every output Target of A has status `OK` **then**
    return (G', P', A)
end if
return $(G, P, \varnothing)$

checks that previously resolved outputs are unchanged:

$$\text{Stable}(P, P') = \bigwedge_{t \in \mathcal{Z}(P)} \left[\sigma'_{e,t} = \texttt{OK} \ \wedge \ \text{Norm}_t(\hat{y}'_{e,t}) = \text{Norm}_t(\hat{y}_{e,t})\right]. \tag{10}$$

Here Norm_t follows the field's value and occurrence contract. Algorithm 4 accepts the retained routes only if this condition and their successful-output checks hold. This is a consistency check on the current admission. Model-based source support does not establish correctness on future patients or provide an independent reference label.

B.5 PROCEDURE IDENTITY AND STREAM UPDATES

In Equation 5, uniqueness depends on operator type and version, input mode and bindings, output Target, parameters, and applicability conditions. Generated names and bookkeeping metadata do not distinguish programs. Updates append distinct alternatives and preserve previously stored procedures. Patient values, source records, intermediate predictions, and execution observations are excluded from the reusable graph. Every retained procedure is executed on the next admission's own records.

Algorithm 5 processes a fixed patient order and the admissions belonging to each patient. Cache commits occur after each admission; training snapshots are saved only after all admissions of a patient have been processed. The checkpoint labels G_p follow the patient-count convention in Section 4.3, rather than the admission index n in Equation 5. Acceptance uses the current admission's Source evidence without accessing $y^\star$. It permits immediate procedural reuse, so training order can affect which programs are available to later admissions. Neither graph growth nor a successful acceptance check implies monotonic extraction-quality improvement.

B.6 SNAPSHOT EVALUATION

Training uses $\rho = 1$ in Algorithm 5. Accepted procedures are retained for later admissions in the same ordered training stream. Earlier predictions are not recomputed after subsequent updates.

Algorithm 5 Stream execution with cross-patient route evolution

Require: Initial graph G_{init}, ordered patients and admissions, retention flag ρ, fixed search and model settings
Ensure: Admission predictions, final reusable graph, and any scheduled training snapshots
$G \leftarrow G_{\text{init}}$; save the initial snapshot if scheduled
for each patient in the fixed order **do**
  for each admission e of this patient in the declared order **do**
    Read permitted Sources x_e; initialize local state and model accounting
    $(P_e, \Delta_e) \leftarrow \textsc{GraphIE}(G, x_e, \mathcal{Q}, \Lambda, \mathcal{B})$
    Record P_e and its execution cost
    if $\rho = 1$ **then**
      Check that G still matches the snapshot used for this admission
      $G \leftarrow \textsc{CommitUniqueRoutes}(G, \Delta_e)$ using Equation 5
    end if
    Discard the local graph, predictions, and memo from reusable execution state
  end for
  Save a snapshot if this patient completes a scheduled training prefix
end for
return recorded predictions, G, and saved snapshots

Table 6: Model-facing roles in ReCIPE. Target reference answers are excluded from every role.

Role	Supplied context	Required output
Compiler	Ranked field definitions; operator and dictionary contracts; cached procedures; resolved-field identifiers and rejection metadata.	A JSON list of executable routes, each with one Target output. The list may be empty.
Extraction operator	The output field definition, route-specific extraction instructions, and bound Source text.	Documented values with exact contiguous source quotations and occurrence information.
Verifier	Proposed routes, trial outputs, relevant intermediate predictions, complete relevant Source records, and applicable dictionary evidence.	One verdict per route: `SUPPORTED`, `REFUTED`, or `UNKNOWN`.

Snapshot evaluation sets $\rho = 0$ and starts every admission from the same snapshot G_p (or the physician-designed graph G_{ref}). GraphIE first executes the stored routes, including any model-based extraction operators, and then permits candidate retrieval, route proposal, and verification for eligible unresolved Targets. A `NOT_FOUND` status from Algorithm 3 can therefore trigger the discovery loop under Λ. Accepted routes update only the admission-local graph $\widetilde{G}_e$ and can contribute to the current admission's predictions. After the predictions and costs are recorded, all local additions and updates are discarded; the next admission starts from the unchanged stored snapshot. For G_0, the initial graph is empty and route construction occurs within the current admission's discovery budget. Fields still unresolved when execution and discovery stop retain their returned missing or failure status. This protocol evaluates how prior procedural experience supports extraction with local discovery, without accumulating procedures across test admissions. The scoring and cost-accounting rules are collected in Appendix E. G_{80} is the graph learned after processing the full training partition.

B.7 Agent Inputs and Output Contracts

The compiler, extraction operator, and verifier use separate instructions and structured output contracts. Their input boundaries determine which information can influence a reusable procedure. Table 6 summarizes these boundaries.

Compiler constraints. The compiler receives definitions and execution metadata without patient values or clinical text. It must use declared fields, preserve acyclicity, and supply a procedure for any unresolved intermediate it introduces. The instructions permit deterministic, hybrid, and model-based extraction routes, while excluding patient identifiers, answer values, and patient-specific constants from reusable parameters. Proposals add alternatives to cached procedures.

Extraction and verification constraints. Extraction instructions require documented facts, all eligible repeated occurrences, and exact quotations. They preserve negation, uncertainty, subject identity, and event time. The verifier checks the procedure and its complete output collection, including meaning, coverage, types, timing, units, joins, and input use. Executability, a valid quotation, or agreement with an earlier prediction is insufficient by itself for acceptance. Both roles treat clinical Source content as data rather than instructions. Static validation and the component-level acceptance rules remain enforced by GraphIE, as specified in Appendix B.4.

C MIMIC-IV-GRAPH CONSTRUCTION AND FIELD DEFINITIONS

This appendix describes the benchmark introduced in Section 3.1. Scoring rules and exclusions appear in Appendix E.3.

C.1 COHORT CONSTRUCTION AND PATIENT SPLIT

We build MIMIC-IV-Graph from MIMIC-IV (Johnson et al., 2024), MIMIC-IV-ED (Johnson et al., 2023a), and MIMIC-IV-Note (Johnson et al., 2023b), which are hosted on PhysioNet (Pollard et al., 2026). Eligible admissions have linked emergency-department and ICU stays, a nonempty discharge summary, and a nonempty radiology report. After excluding previously designated development and pilot patients, deterministic sampling selects 2,500 patients using source availability and fixed patient-identifier hashes. The patient-disjoint split contains 2,000 training patients with 2,487 admissions and 500 test patients with 620 admissions. This cohort provides linked textual and tabular evidence within each admission; its eligibility criteria restrict evaluation to patients with these record types.

C.2 FIELD COVERAGE AND OCCURRENCE IDENTITY

Sources include complete discharge and radiology notes, diagnosis and procedure codes, laboratory results, and service and transfer records. Targets span structured clinical variables, note sections, medication attributes, and documented clinical statements. Two Targets are scalar and 41 are sequences that retain distinct occurrences, including repeated values. Record identifiers preserve correspondence across fields; note-derived occurrences use character spans in the original note, and medication attributes share their instruction's identifier.

C.3 REFERENCE CONSTRUCTION

We construct references directly from the original tables and notes without model-generated labels. Thirteen Targets use recorded table values or projections through code dictionaries. The remaining 30 comprise nine note-section fields, 20 fields for documented entries and attributes, and one medication dose-difference field. Fixed rules identify section boundaries, extract entries, and normalize declared aliases, numbers, and timestamps. Radiology negation and uncertainty fields retain complete statements selected by specified lexical cues. The dose-difference field requires a unique medication-name match within the same note and an eligible single dose in mg on each side; ambiguous matches are excluded. Its value describes documented quantities, without inferring the dose actually administered. These references measure agreement with the specified extraction rules; assessing clinical completeness beyond those rules requires separate annotation.

C.4 SHARED DICTIONARIES AND SOURCE ACCESS

ICD and HCPCS mappings come from the released dictionary tables. The ETC dictionary contains unambiguous code–description pairs from the full ED medication-reconciliation table. It is provided as a shared auxiliary resource before training. Methods may acquire reusable state from training Sources. The permitted source fields and their types are listed below; Target references remain reserved for offline scoring.

C.5 Reproducible Dataset and Dictionary Construction

Inputs and implementation. The construction scripts are `scripts/preprocess/build_dataset.py` and `scripts/preprocess/build_dictionaries.py`. They accept the released CSV or compressed CSV.GZ tables from the three dataset versions listed above. Dataset construction uses Python 3.12 or later, DuckDB, and PyYAML; dictionary construction uses only the Python standard library. Field contracts are read from `data/MIMIC-IV-Graph/fields/` in the repository. The scripts construct values from the raw tables and notes without model calls or previously generated answers.

Patient exclusions and deterministic selection. Eligibility is computed by joining admissions, ICU stays, ED stays, discharge notes, and radiology reports on both patient and admission identifiers. Note records must have nonempty text after whitespace trimming. The exclusion rule starts with all distinct patients in the admissions table, sorted numerically, and shuffles them using Python's `random.Random(20260301)`. For N such patients, let $M = \lfloor 4N/5 \rfloor$ and $D = \lfloor M/10 \rfloor$. The first D shuffled patients form a previously reserved subset and are excluded. The script also excludes two pilot samples: 30 patients from positions D through $M - 1$ and 10 from positions M through $N - 1$, using zero-based positions. Each pool is numerically sorted before sampling with seeds 20260906 and 20260907, respectively.

The remaining eligible patients are ranked by the SHA-256 digest of the UTF-8 string

```
routegraph-cohort-v1|20260301|<domain>|<subject_id>
```

with the numeric patient identifier breaking any digest tie. The domain `selection` selects the first 2,500 patients. These patients are then ranked again using the domain `formal-split`; the first 2,000 form the training set and the remaining 500 form the test set. All eligible admissions of each selected patient are retained and ordered by numeric admission identifier within that patient. This procedure depends on source availability and patient identifiers, without consulting Target values.

Output format. The dataset builder writes `train.csv` and `test.csv`, with one admission per row and 64 columns. The first three columns are `split`, `subject_id`, and `hadm_id`, followed by the 18 Source identifiers and 43 Target identifiers, each group sorted by field identifier. Missing scalar values are empty cells. Sequence cells contain JSON lists of `record_id` and `value` objects, with empty sequences represented by `[]`. Record identities and note offsets preserve repeated occurrences and cross-field alignment.

Dictionary export. The dictionary builder writes `icd_diagnoses.json`, `icd_procedures.json`, `hcpcs.json`, and `etc.json`. ICD keys concatenate the original code and version with a vertical bar; HCPCS and ETC use their original code strings. The ETC export scans the full medication-reconciliation table, skips empty codes or descriptions, and retains a code only when it has exactly one distinct description. The dataset builder reads its required official dictionaries in memory; the separate export supplies the reusable lookup files.

Reconstruction commands. From the repository root, the CSV files and dictionary files can be generated with:

```
uv run python scripts/preprocess/build_dataset.py \
  --mimic /path/to/MIMIC-IV \
  --note /path/to/MIMIC-IV-Note \
  --ed /path/to/MIMIC-IV-ED \
  --output /path/to/new-output

uv run python scripts/preprocess/build_dictionaries.py \
  --mimic /path/to/MIMIC-IV \
  --ed /path/to/MIMIC-IV-ED \
  --output /path/to/new-output/dictionaries
```

Both scripts reject overwriting existing output files with the same names.

C.6 Complete Source and Target Catalog

Tables 7 and 8 list all 18 Sources and 43 Targets. All Sources are sequences. The two scalar Targets are marked with an asterisk; the other 41 Targets are sequences. The Target inventory contains 11 Enum, 4 Numeric, 2 Datetime, and 26 Text fields. Each machine-readable field definition additionally specifies its scope, normalization, units, and permitted categorical aliases. Medication attributes retain their enclosing instruction's occurrence identity, and note-derived occurrences retain their original-note spans.

Table 7: Source field catalog. All fields have sequence cardinality.

Field identifier	Meaning	Type
s.current_service	Current hospital service	Enum
s.diagnosis_code	Inpatient diagnosis ICD code	Text
s.diagnosis_version	Inpatient diagnosis ICD version	Enum
s.discharge_text	Full discharge note text	Text
s.ed_diagnosis_code	ED diagnosis ICD code	Text
s.ed_diagnosis_version	ED diagnosis ICD version	Enum
s.ed_etc_code	ED medication therapeutic class code	Text
s.hcpcs_code	Recorded HCPCS code	Text
s.icu_los	ICU length of stay	Numeric
s.lab_50912_text	Creatinine recorded result	Text
s.lab_50912_unit	Creatinine result unit	Text
s.procedure_code	Inpatient procedure ICD code	Text
s.procedure_version	Inpatient procedure ICD version	Enum
s.radiology_text	Full radiology report text	Text
s.service_time	Service transfer time	Datetime
s.transfer_careunit	Transfer care unit	Text
s.transfer_intime	Transfer entry time	Datetime
s.transfer_outtime	Transfer exit time	Datetime

Table 8: Target field catalog. An asterisk marks scalar cardinality; all other fields are sequences.

Field identifier	Meaning	Type
t.diagnosis_title	Inpatient diagnosis description	Text
t.discharge_location*	Hospital discharge location	Enum
t.ed_diagnosis_title	ED diagnosis description	Text
t.ed_etc_description	ED medication therapeutic class description	Text
t.hcpcs_description	HCPCS procedure description	Text
t.hospital_expire_flag*	In-hospital death indicator	Enum
t.icu_first_careunit	First ICU care unit	Enum
t.icu_intime	ICU admission time	Datetime
t.icu_last_careunit	Last ICU care unit	Enum
t.icu_outtime	ICU discharge time	Datetime
t.lab_50912_number	Creatinine numeric result	Numeric
t.med_admission.dose	Admission medication documented dose	Numeric
t.med_admission.drug	Admission medication name expression	Text
t.med_admission.frequency	Admission medication frequency	Enum
t.med_admission.orders	Admission medication instructions	Text
t.med_admission.route	Admission medication administration route	Enum
t.med_admission.unit	Admission medication dose unit	Enum
t.med_discharge.dose	Discharge medication documented dose	Numeric
t.med_discharge.drug	Discharge medication name expression	Text

Continued on the next page

Table 8 (continued)

Field identifier	Meaning	Type
t.med_discharge.frequency	Discharge medication frequency	Enum
t.med_discharge.orders	Discharge medication instructions	Text
t.med_discharge.route	Discharge medication administration route	Enum
t.med_discharge.unit	Discharge medication dose unit	Enum
t.medication.documented_dose_delta_mg	Documented discharge minus admission medication dose (mg)	Numeric
t.note.allergens	Explicitly listed allergy substances	Text
t.note.course_problems	Explicit hospital-course problem headings	Text
t.note.diagnosis_items	Documented discharge diagnosis expressions	Text
t.note.past_history	Documented past-history statements	Text
t.previous_service	Previous hospital service	Enum
t.procedure_title	Inpatient procedure description	Text
t.radiology.measurements	Statements containing documented imaging measurements	Text
t.radiology.negative	Statements with explicit negation cues	Text
t.radiology.statements	Radiology assessment statements	Text
t.radiology.uncertain	Statements with explicit uncertainty cues	Text
t.section.allergies	Allergy history section text	Text
t.section.condition	Discharge condition section text	Text
t.section.course	Hospital course section text	Text
t.section.diagnosis	Discharge diagnoses section text	Text
t.section.disposition	Discharge disposition section text	Text
t.section.medications_admission	Medications on admission section text	Text
t.section.medications_discharge	Discharge medications section text	Text
t.section.pmh	Past medical history section text	Text
t.section.radiology	Radiology assessments section text	Text

D BASELINES

Table 1 compares eight baselines spanning direct extraction, retrieval, workflow execution, and agent-based planning. Each comparison uses a task-specific adaptation to the common Source/Target interface, with the aim of following the original method as closely as possible. All methods receive the same Source fields, public field definitions, dictionaries, and operator interfaces; Target references are reserved for offline evaluation. LLM-based methods use the shared backbone specified in Appendix E.

D.1 SHARED ADAPTATION PROTOCOL

Outputs follow the same field types, normalization rules, and occurrence identities. Sequences retain source-record identifiers or source-character spans, and medication attributes share the identity of their enclosing instruction. Extraction requests support batches of up to six Target fields, with correction and subdivision when outputs fail validation. Execution failures remain explicit at the field level. Deterministic parsing, normalization, and dictionary lookup are provided as execution utilities. Compilation costs follow the accounting rules in Appendix E.4.

D.2 DIRECT EXTRACTION

Direct LLM. Direct LLM uses the language model to extract requested values from Source records and Target definitions through direct prompting. It provides a reference for the shared backbone's extraction ability before introducing evidence retrieval, executable planning, or reusable procedures.

D.3 RETRIEVAL-BASED

Hybrid-RAG. Hybrid-RAG addresses evidence selection by combining lexical matching with text-similarity signals before passing retrieved records to the language model. We include it to compare retrieval-guided extraction with ReCIPE's use of explicit executable dependencies across clinical fields, encoded in RouteGraph.

HippoRAG 2. HippoRAG 2 (Gutiérrez et al., 2025) addresses non-parametric continual learning by organizing passages and their relationships for factual, associative, and sense-making retrieval. The original framework combines graph-based retrieval through personalized PageRank with passage integration and LLM-assisted retrieval. Its adaptation provides a comparison with graph-organized evidence retrieval when relevant clinical information is distributed across records.

D.4 WORKFLOW-BASED

CAESURA. CAESURA (Urban & Binnig, 2024) addresses natural-language querying over heterogeneous tables, text, and images. The original system uses language models to construct execution plans that combine database operations with model-based processing. We compare against this planning-oriented approach because clinical field extraction likewise requires selecting sources and composing operations across textual and tabular evidence.

EVAPORATE-CODE+. EVAPORATE (Arora et al., 2023) constructs queryable tables from heterogeneous documents using language models. Its CODE+ formulation synthesizes extraction functions and combines their outputs through weak supervision to improve the quality–cost tradeoff. Our task-adapted comparison examines reusable extraction code as an alternative way to reduce repeated model processing across admissions.

OmniTQA. OmniTQA (Shahbazi et al., 2026) addresses questions whose answers depend on both explicit relational columns and information implicit in free-form text. The original framework represents relational and semantic operations within a common query plan and optimizes their execution. It is relevant to our setting because clinical Targets can require both deterministic table processing and semantic interpretation of notes.

DocETL. DocETL (Shankar et al., 2025) addresses complex document-processing tasks for which a single model operation may miss relevant information. The original system expresses processing as declarative pipelines and uses agentic rewrites to decompose operations or data and evaluate alternative plans. We include its adaptation to compare document-workflow organization for extraction and aggregation over long clinical records.

D.5 AGENT-BASED

EHRAgent. EHRAgent (Shi et al., 2024) addresses multi-table reasoning over electronic health records through language-model-generated code and tool use. The original agent combines medical knowledge, data-access planning, and iterative code execution with feedback. It provides a clinically grounded comparison for agent-based planning and execution under the shared field-extraction protocol.

E EXPERIMENTAL SETTINGS, METRICS, AND COST ACCOUNTING

This appendix collects the evaluation protocol, model and search settings, scoring rules, and cost definitions for Experiments 1–3. Benchmark construction is described in Appendix C; baseline mechanisms appear in Appendix D. The separate physician-review analysis is specified in Appendix G.

E.1 MODEL DEPLOYMENT AND EVALUATION PROTOCOL

All LLM-based methods use the Qwen3.5-27B backbone (Qwen Team, 2026) deployed locally on eight NVIDIA A800 GPUs. For field-definition retrieval, we use a locally deployed Qwen3-Embedding-4B encoder. Its inputs contain only public field-definition metadata. The evaluation

Table 9: Model and route-discovery configuration used for ReCIPE during training and snapshot evaluation. Only training retains accepted routes across admissions.

Setting	Value
Definition encoder	Qwen3-Embedding-4B (local)
Embedding dimensionality	2,560
Definition similarity	Cosine similarity after ℓ_2 normalization
Sampling	Temperature 0.0; top-p 1.0
Generation seed	20260301
Patient-order seed	20260923
Maximum output length	16,384 tokens per request
Requested thinking budget	2,048 tokens
Structured responses	JSON schema; one format-repair attempt
Discovery rounds / initial candidate width	1 / 8
Proposal batches per admission	At most 8
Targets / routes per proposal batch	At most 6 / 8
Cached-miss revisit schedule	Approximately one in 16 admissions; at most one proposal batch
Admission model budget	256 calls; 2,000,000 tokens
Route workers / simultaneous model requests	1 / 1

unit is a hospital admission. The patient-disjoint split contains 2,000 training patients and 500 test patients, corresponding to 2,487 and 620 admissions. Methods receive the public field definitions, authorized dictionaries, and Source records; Target references are available only to the evaluator.

In Experiments 1–3, GraphIE initializes each test admission from the stored graph snapshot for the evaluated condition. Route discovery is enabled within each admission, alongside execution of the stored routes and their model-based extraction operators. Accepted routes and other local updates are discarded after that admission; the stored snapshot remains unchanged across test admissions. The same held-out admissions are used for the snapshot comparison and the physician-designed graph comparison. The snapshot labels follow Equation 7, with G_0 preceding learning and G_{80} denoting the graph learned from the full training partition.

E.2 Model and Route-Learning Configuration

Table 9 lists the model and search settings used for ReCIPE in our experiments. The discovery settings apply during training and snapshot evaluation; accepted routes are retained across admissions only during training. The proposal limit bounds search attempts; it does not require every admission to produce a new route or resolve every Target.

E.3 Scoring Eligibility and Aggregation

Eligible admission–Target pairs. The evaluator scores pairs with nonempty references; missing scalar references and empty reference sequences are excluded. Failed predictions remain in the evaluation whenever the reference is valid. An empty reference cannot be interpreted as a clinically verified negative. The type-level scores give equal weight to fields with eligible references, rather than pooling all occurrences across fields.

Enum. Predicted and reference occurrences are aligned by their declared occurrence identities; scalar fields use a single identity. For an eligible admission–field pair with m reference occurrences, each occurrence contributes weight $1/m$ to the class-specific true-positive, false-positive, and false-negative counts. Missing, extra, or incorrectly classified occurrences contribute the corresponding errors. Counts are accumulated across admissions, and class F1 scores with nonzero denominators are averaged within each field. The Enum score is the mean of these field-level macro-F1 scores.

Numeric and Datetime. Exact match compares the complete normalized scalar or occurrence collection for each eligible admission–field pair. For a sequence, both occurrence identities and normalized values must match under the field's declared ordering. A partial collection does not receive partial EM credit. Numeric normalization preserves comparators and applies the declared

decimal precision with half-even rounding. Datetime normalization follows the declared date or timestamp precision. Scores are averaged across eligible admissions within each field and then equally across fields of the same type.

Text. Sequence values are joined with newline separators in their supplied order before text scoring. Text quality is evaluated with segmented BERTScore-F1 (Zhang et al., 2020), using a locally loaded RoBERTa-large encoder (Liu et al., 2019). The scorer uses layer 17, with IDF weighting and baseline rescaling disabled. Predictions and references are tokenized without special tokens and divided into consecutive segments of at most 450 tokens. All prediction–reference segment pairs are scored. For each predicted segment, the maximum precision over reference segments is retained; these maxima are averaged using predicted segment lengths as weights. Recall is computed symmetrically using reference segment lengths. The final pair score is the harmonic mean of these aggregated precision and recall values. An empty prediction against a nonempty reference receives zero. Pair scores are averaged first within each field and then across Text fields.

Overall. Equation 6 combines the four unrounded type scores with equal weights. A zero component yields an Overall score of zero; a missing component leaves Overall unavailable. The optional fact-based Text metric in the evaluator is not a component of the reported Overall score. The physician-review statistics are evaluated separately.

E.4 Token, Call, and Runtime Accounting

The cost denominator is a test admission, including admissions with incomplete or failed predictions. Input and output tokens are summed without counting separately reported reasoning tokens a second time. API-call counts include retries and format-repair requests, with repeated log entries for the same request deduplicated. Unknown token usage is retained as missing rather than treated as zero. One-time baseline program compilation is recorded separately, and this cost is excluded from the reported per-admission costs. ReCIPE training and RouteGraph construction costs are separate from test-time extraction costs, which include admission-local discovery and verification.

Per-admission elapsed time covers case-runtime preparation, route execution, route discovery, verification, and retries within the extraction attempt. We report mean elapsed time, the arithmetic mean of these durations over all test admissions. Incomplete and timed-out admissions remain in the denominator and contribute their recorded elapsed time. Figure 3 reports this mean under the 40 s per-admission budget specified in Section 4.3. The field-definition index is cached and reused during training and snapshot evaluation. Index preparation and evaluation scoring are accounted for separately from per-admission extraction.

F Physician-Reviewed Reference and Learned-Procedure Analysis

F.1 Graph Construction and Structural Comparison

Two physicians jointly specify the reference extraction graph. We transcribe their diagram into executable operators and parameters, and the physicians manually review the resulting executable specification and confirm its clinical extraction logic. Reference validation consists of this manual expert review. The comparison evaluates the complete extraction programs produced by physician specification and source-grounded route learning by ReCIPE. The analysis compares the executable files `G_ref.yaml` and `G_80.yaml`, aligning their projected field-to-field dependencies by input and output fields. Matching field pairs establish structural correspondence; route identity also depends on operator type and version, input mode and bindings, parameters, and applicability conditions.

Structure. Table 10 distinguishes projected field-to-field dependencies from the Field–Operator arcs of the executable graph. A route is a configured operator instance represented by one graph node; internal steps of a composite operator do not add graph nodes. Projected dependencies retain distinct direct input–output field pairs, whereas the Source-to-Target paths in Table 2 may traverse multiple routes. Neither graph declares explicit applicability guards. In G_{80}, the four targets with alternative routes are hospital mortality and the three radiology attributes. Both graphs structurally

cover every Target and have the same maximum route depth. Their extraction quality therefore depends on the evidence and transformations specified by the routes within that structure.

Table 10: Structure of the two executable graphs. Projected dependencies connect each route's input fields directly to its output field, omit operator nodes, and merge duplicate field pairs across routes. Route depth counts route nodes on the longest Source-to-Target path.

Quantity	G_{ref}	G_{80}
Source / Target fields	18 / 43	18 / 43
Routes (operator instances)	43	47
Projected field-to-field dependencies	65	69
Field–Operator arcs	108	116
Targets with multiple routes	0	4
Structural Target coverage	100%	100%
Maximum route depth	4	4

F.2 Field-Level Scores and Paired Outcomes

Using the unrounded scores underlying Table 3, the overall advantage of G_{80} over G_{ref} is 0.0656, with the largest type-level differences in `Datetime` (0.1258) and `Enum` (0.0898). Table 11 reports selected fields, including the mortality case discussed in the main text and the ICU diagnostic below. The mortality and discharge-location fields together contribute 4.77 of the 8.98 percentage-point Enum gain (53.1%). For mortality, the paired outcomes are 75 admissions correct under both graphs, 271 correct only under G_{80}, two correct only under G_{ref}, and 272 correct under neither. Of the 271 improvements, the recorded producing-route identifiers for mortality are R46 alone in 264, R45 and R46 in six, and R45 alone in one. These are exact paired outcomes, distinct from the Macro-F1 scores in the table. The recorded identifiers name the routes that directly produced the mortality prediction; the complete execution trajectory also includes upstream operations and any unsuccessful attempts. The three medication fields have 27, 13, and seven improvements among 449, 491, and 342 eligible admissions, respectively, and no paired regressions. Admissions can contribute to more than one field, so these improvements must not be summed as a count of unique patients.

Table 11: Selected field scores from the original saved predictions. Enum uses Macro-F1; Numeric and Datetime use normalized EM. Eligible admissions are those with valid gold values.

Field	Type	Eligible	G_{ref}	G_{80}
Hospital mortality	Enum	620	0.4869	0.8306
Discharge location	Enum	614	0.0538	0.2347
First ICU care unit	Enum	620	0.7539	0.8442
Last ICU care unit	Enum	620	0.8792	0.9992
ICU admission time	Datetime	620	0.8694	0.9952
ICU discharge time	Datetime	620	0.8694	0.9952
Admission medication dose	Numeric	449	0.9354	0.9955
Discharge medication dose	Numeric	491	0.9674	0.9939
Documented dose change	Numeric	342	0.9795	1.0000
Creatinine	Numeric	618	0.9515	0.9515

F.3 Executable Category Coverage: A Post-Hoc ICU Diagnostic

The ICU fields illustrate how operator parameters determine which clinical records an executable route can use. The learned graph includes `Coronary Care Unit (CCU)` in its ICU category filters, while the reference filters omit this category. Both graphs use deterministic operators for these fields. The reference's four ICU routes each contain a composite sequence with a category filter and an interval-selection step. We add `Coronary Care Unit (CCU)` to both category gates in each route, making eight list edits in an in-memory copy. All dependencies, operator types, priorities, and other parameters remain fixed. We then execute only these four deterministic targets,

without model calls. The unmodified graphs reproduce their saved ICU outputs on all 620 admissions, and the modified reference matches the replayed G_{80} on every admission for each of the four fields. For each timestamp field, the number of correct admissions increases from 539 to 617. This controlled change identifies category coverage in the executable parameters as the source of the ICU difference. Alongside the additional mortality evidence route, it illustrates the two aspects of a reusable procedure evaluated here: the choice of evidence inputs and the operations and parameters applied to them. The category edit was selected after inspecting the test results and is used only for this diagnostic. The published comparison always evaluates the original, physician-reviewed reference graph.

Score aggregation. Overall is computed from the four full-precision type scores, with Text evaluated using segmented BERTScore-F1. Failed predictions remain in the denominator whenever gold is valid. The complete scoring rules appear in Appendix E.3.

G Physician Review Protocol and Interface

This appendix supplements the blinded physician review in Section 4.5. It specifies the sampling unit, review materials, aggregation rule, and uncertainty calculation.

G.1 Disagreement Sampling and Blinding

The review uses oncology records from the collaborating cancer center. An entry is eligible when ReCIPE and the center-provided Reference produce different values. The sample contains 100 entries from each of the four field types, for 400 entries in total. Reference values are prepared through manual abstraction of requested fields from clinical records and simple regular-expression (regex) rules for matching text patterns.

For each entry, the two candidate values are randomly assigned to labels A and B. Five credentialed physicians independently review the field name, definition, candidate values, and relevant excerpts from the original records. Each selects one of four judgments: only A is correct, only B is correct, both are correct, or neither is correct. Method identities remain hidden during review and are restored only for analysis.

G.2 Review Interface

Figure 5 gives an anonymized schematic of the information presented to reviewers. It contains no patient example or method identity.

Field [Target field name]

Definition [Meaning, scope, units, and output requirements]

Candidate A	**Candidate B**
[First blinded value]	*[Second blinded value]*

Source evidence
[Relevant excerpts from the original records]

Select one judgment

○ Only A is correct	○ Only B is correct
○ Both are correct	○ Neither is correct

Figure 5: **Schematic of the blinded review interface.** The displayed information comprises the field and its definition, candidates A and B, relevant Source excerpts, and four judgment options. Bracketed text denotes placeholders; this is a schematic, not a captured patient review.

G.3 Voting, Inter-Rater Agreement, and Confidence Intervals

Item-level outcome. The category receiving the most votes determines the final outcome. This is plurality voting and does not require an absolute majority. No ties for the highest count occur among

the 400 reviewed entries. The analysis requires complete ratings from all five physicians; missing ratings or a tied highest count are flagged rather than resolved arbitrarily.

Mean pairwise inter-rater agreement. Agreement (%) measures how often pairs of the five physicians select the same one of the four judgment options for an entry. Let n_{ic} be the number of physicians assigning category c to entry i. The calculation uses their independent ratings before plurality-vote aggregation. For a set of reviewed entries D, the reported percentage is

$$A_D = \frac{100}{|D|} \sum_{i \in D} \frac{\sum_{c=1}^{4} n_{ic}(n_{ic} - 1)}{5 \cdot 4}. \tag{11}$$

Equivalently, each entry has $\binom{5}{2} = 10$ distinct physician pairs; a pair agrees when both physicians select the same option. The reported percentage averages this fraction equally over entries within each field type.

Paired correctness difference. Let a, b, c, and d count entries judged correct only for ReCIPE, only for Reference, for both, and for neither, respectively. The paired difference is $\Delta_D = (a - b)/(a + b + c + d)$. The confidence interval uses an entry-level percentile bootstrap stratified by field type. Each draw samples with replacement within each type, retaining that type's sample size and each entry's aggregated outcome. The outcomes contribute $+1$, -1, 0, and 0 to the paired difference, respectively. The analysis uses 10,000 draws with seed 20260914 and takes the 2.5th and 97.5th percentiles.

Scope of inference. The estimated difference concerns sampled disagreements between the two workflows. It does not estimate their accuracy difference over all hospital records. The bootstrap quantifies entry-sampling uncertainty conditional on the physician ratings and the fixed aggregation rule; it does not resample physicians. Repeated entries from the same patient are not treated as patient clusters by this bootstrap.