# OZONE: OBJECT-ORIENTED OPTIMIZATION OF AGENTIC WORKFLOW FOR SCIENTIFIC SURVEY GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We introduce OZONE, an object-oriented agentic workflow optimization framework for scientific survey generation. Survey writing comprises tasks that share common requirements but differ in their objectives, operator compositions, and evaluation criteria. OZONE represents the search space with predefined abstract classes and task-specific subclasses that inherit common contracts. The optimizer constructs and extends operator implementations under these contracts, assigns execution units to subclasses based on their operators and outputs, and connects the units into a directed acyclic graph. We design local rubrics for intermediate outputs and a global rubric for completed surveys, as improvements to individual tasks do not always improve overall survey quality. Local diagnosis guides changes to unit implementations and their connections, while the global score provides a basis for comparing candidate workflows. At inference time, OZONE supports runtime editing guided by a rubric proxy and refinement through review-based revision. We introduce CROFTBench, a benchmark of recent CS survey topics that extends evaluation to figures and tables. With GPT-4o-mini and inference refinement, OZONE achieves 72.2 on SurveyBench, 46.02 on Bloom-Eval, and 43.34 on CROFTBench.

## 1 INTRODUCTION

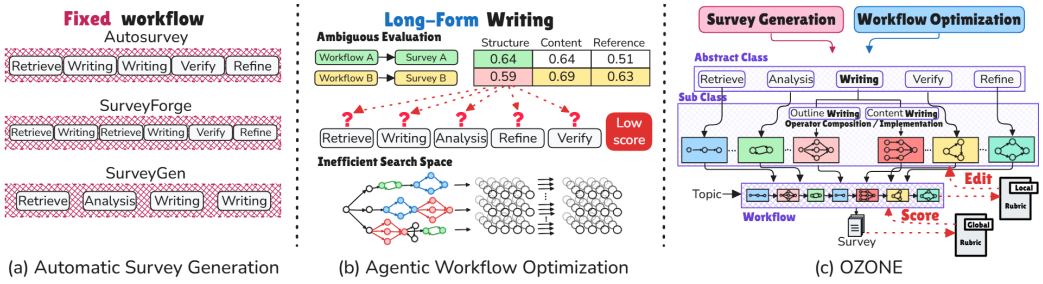



Figure 1: Overview of the proposed method.

The growing volume of scientific publications creates a need to systematically organize and synthesize the literature. Survey papers guide future research by comparing existing approaches and identifying open questions. Survey writing requires cross-paper comparison and synthesis grounded in representative evidence, beyond summaries of individual papers. Citations connect claims to supporting sources, while figures and tables present structured comparisons of methods and findings. Producing a coherent, evidence-grounded survey requires coordinating literature retrieval, outline construction, content generation, and refinement (Wang et al., 2024; Yan et al., 2025).

Existing survey-generation approaches organize literature retrieval, outline planning, and section writing into pipelines, or use a planner to select the modules needed during writing (Wang et al., 2024; Yan et al., 2025; Chao et al., 2025). Memory and iterative refinement improve the evidence,



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.

plan, and draft of the survey being written (Yan et al., 2025; Chen et al., 2025), while the procedure within each stage and the connections between stages are fixed at design time. Fully automatic survey generation still faces challenges in citation quality and critical analysis (Bao et al., 2025), and generated surveys receive lower scores for knowledge integration than for structural organization (Zhang et al., 2026a). Because these weaknesses may arise from the scope of retrieved evidence, the comparative structure built during analysis, or the synthesis performed during writing, addressing them requires adapting not only individual prompts or modules but also the procedure within each stage and the dependencies between stages. We therefore treat survey generation as a workflow optimization problem, where both task implementations and their composition can be searched.

Agentic workflow optimization searches over prompts, operator compositions, and topologies to automate workflow construction, typically evaluated on short-answer benchmarks (Zhuge et al., 2024; Hu et al., 2025; Zhang et al., 2025b;a; Zhou et al., 2026). For survey generation, aggregate document scores support comparison across candidate workflows but provide limited guidance on unmet writing requirements and the implementation edits that address them (Song et al., 2024). Execution feedback, trajectory reflection, and local rewards guide optimization (Zhang et al., 2025b; Agrawal et al., 2026; Wu et al., 2026), but in a long document a local improvement can introduce redundancy or inconsistency elsewhere. Modular search spaces and experience memory support structured search and reuse (Shang et al., 2025; Zhao et al., 2026; Liu et al., 2026; Zhang et al., 2026b), yet their structure follows module interfaces or operator similarity rather than the writing requirements that tasks share.

Long-form survey generation consists of tasks that share requirements but differ in their immediate objectives. Outline writing and content writing both require coherent organization, yet the former determines the scope and hierarchy of the discussion, whereas the latter compares and synthesizes evidence. A workflow representation should preserve the requirements these tasks share while letting their implementations and evaluation criteria specialize, rather than applying one implementation to both or treating them as unrelated. Object-oriented design provides this structure: an abstract class fixes the shared role and input–output contract, and subclasses inherit it while specializing their implementations. These requirements also serve as criteria for intermediate outputs. Meeting them does not guarantee a better survey, so document-level quality must be assessed separately. We investigate how a search representation built on shared requirements and task-specific specialization, evaluated at both task and document levels, affects survey-generation workflow optimization.

We propose OZONE (Object-Oriented Optimization of Agentic Workflows), which represents the workflow search space with this class structure. OZONE defines abstract classes for five writing roles (retrieve, analyze, write, verify, and refine) and a predefined space of task-specific subclasses that inherit their contracts. The optimizer constructs and extends operator implementations under the inherited contracts, and assigns each execution unit to a subclass according to its operators and outputs. Units are connected as a directed acyclic graph (DAG); the implementation of each unit and the composition of units are the search targets. Subclass-specific local rubrics check whether intermediate outputs meet the writing requirements of their task and identify units to revise, while a global rubric scores completed surveys to compare candidate workflows. Each round initializes a workflow from the class hierarchy and then repeatedly edits the implementation or composition of units according to local diagnosis. We compare local and global scores before and after each edit to assess its effect, and record the outcome by subclass and edit type for later search. At the end of a round, surveys generated by the winning workflow are reviewed, and their common weaknesses are passed to the next round as feedback. The selected workflow is reused on unseen topics without further search. We further introduce CROFTBench, a benchmark of 10 computer science (CS) topics selected from recent surveys that extends evaluation to figure and table generation.

Our contributions are fourfold.

- We propose OZONE, an object-oriented framework that optimizes survey-generation workflows over abstract classes for shared requirements and subclasses for task-specific objectives.
- We design and optimize survey-specific global and local rubrics to assess document quality, subclass-level intermediate outputs, and the effects of workflow edits.
- We introduce CROFTBench, a benchmark based on recent literature that includes figure and table generation and evaluation, extending the scope of survey-generation assessment.
- With GPT-4o-mini, OZONE achieves 72.2 on SurveyBench, 46.02 on Bloom-Eval, and 43.34 on CROFTBench.

## 2 Related Work

**Automatic Survey Generation** Survey-generation approaches combine literature retrieval with outline-based writing and use memory to improve consistency across sections (Wang et al., 2024; Yan et al., 2025). Evolving plans and dynamic module selection support adaptation during writing (Chen et al., 2025; Chao et al., 2025), while quality-aware retrieval incorporates source-paper quality into evidence selection (Bao et al., 2025). These studies refine evidence selection and the writing process. OZONE focuses on searching over task-specific operator implementations and workflow composition within the writing requirements shared across tasks.

**Workflow Optimization** Automated design of agentic systems searches over different representations. ADAS and AFlow generate and revise agent code (Hu et al., 2025; Zhang et al., 2025b), GPTSwarm optimizes edge connectivity in a computational graph (Zhuge et al., 2024), MASS optimizes prompts and topologies in stages (Zhou et al., 2026), and MaAS samples a query-dependent architecture from an agentic supernet (Zhang et al., 2025a). Modular design spaces and abstraction operators organize the search space (Shang et al., 2025; Zhao et al., 2026), and hierarchical experience supports the reuse of execution strategies (Fan et al., 2025). OZONE represents shared survey-writing requirements and task-specific specialization through a predefined class hierarchy, and optimizes subclass implementations and DAG composition under inherited contracts.

**Evaluation and Feedback.** Fine-grained evaluation defines criteria according to the skills required by each instruction (Ye et al., 2024), and agentic evaluation uses hierarchical requirements and intermediate feedback (Zhuge et al., 2025). Component-level reward optimization connects local rewards to global performance (Wu et al., 2026), while rubric-based model training uses explicit evaluation criteria as rewards (Gunjal et al., 2026). OZONE uses survey-specific local rubrics to diagnose subclass-level intermediate outputs and a global rubric to evaluate completed surveys. The two signals guide workflow edits and candidate comparison, respectively.

## 3 Preliminaries

**Survey Generation Workflow** An abstract class defines a shared role and I/O contract, and a subclass specifies a task-specific specialization under that contract. We represent a survey generation workflow as a directed acyclic graph $W = (V, E)$, where $V = \{v_1, \ldots, v_m\}$ is a set of execution units and $E$ contains their dependencies. Each execution unit $v_i = (\mathcal{C}_i, O_i, P_i)$ contains an operator composition $O_i$ configured by prompts and arguments $P_i$, and is associated with a set of predefined subclasses $\mathcal{C}_i$. All subclasses in $\mathcal{C}_i$ share the unit's parent abstract class $A_i$ and inherit its role and I/O contract. A unit may combine multiple task specializations, and units assigned to the same subclass may have different implementations. Dependencies connect compatible interfaces and specify how outputs pass between execution units.

**Workflow Optimization** Given a topic $x$, the workflow $W$ generates a survey $y = W(x)$. The search space $S$ contains executable workflows with different unit implementations and dependencies within the predefined class hierarchy. Let $R_G$ denote the global rubric and $g(x, y; R_G)$ the scalar score obtained by evaluating survey $y$ for topic $x$ using $R_G$. Our objective is to maximize the average global score over a validation set of topics $D$:

$$\max_{W \in S} G(W), \quad G(W) = \frac{1}{|D|} \sum_{x \in D} g(x, W(x); R_G). \tag{1}$$

## 4 Methodology

OZONE optimizes execution-unit implementations and their dependencies over multiple rounds using rubric feedback. Each round initializes a workflow as the root search node. Each expansion selects a parent from evaluated candidates and uses local rubric feedback to edit its unit implementations or dependencies, producing a child workflow. The child is evaluated against its parent, added to the archive, and its edit outcome recorded in memory for subsequent search. At the end of each round, surveys generated by the workflow with the highest global score are reviewed to identify

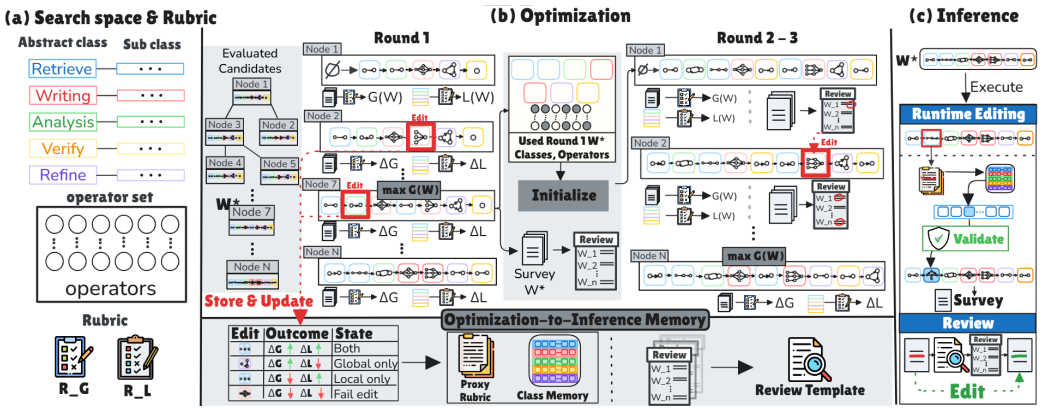



Figure 2: OZONE Architecture Overview. Each search node represents a complete workflow, with colored execution units containing operator compositions. Memory records whether edits improve global quality, local quality, both, or neither. At inference time, class memory guides implementation selection, the proxy rubric evaluates intermediate outputs, and the review guides draft revision.

weaknesses that guide workflow construction and editing in the next round. Figure 2 presents an overview of OZONE; Appendix B provides optimization and inference pseudocode.

### 4.1 OBJECT-ORIENTED SEARCH SPACE

**Abstract Classes and Subclasses** OZONE defines Retrieve, Analysis, Writing, Verify, and Refine as the five abstract classes for survey generation. An abstract class specifies the functional role and I/O contract shared by a group of tasks. The predefined catalog organizes task-specific subclasses under these classes. For example, pool retrieval builds a candidate literature pool, while scoped retrieval selects evidence for a section or subsection. Within Refine, synthesis refinement adds cross-paper comparisons, while content refinement addresses issues in the draft. A unit may perform both tasks and receive both subclass assignments.

To construct a unit $v_i$, the optimizer selects an abstract class $A_i$ and composes operators under its contract. Subclass assignment uses the registered output types and traces the producing operators through the output bindings. Let $B_i$ denote these bindings and the types. We define

$$\mathcal{C}_i = \Phi_{\mathcal{H}}(A_i, O_i, B_i), \qquad \mathcal{C}_i \subseteq \mathcal{C}(A_i), \tag{2}$$

where $\mathcal{H}$ is the predefined hierarchy and $\mathcal{C}(A_i)$ contains its subclasses under $A_i$. The assignment also includes specializations whose declared operators occur in the unit and whose output types are registered by the unit. Units with multiple assignments use the shared criteria of their abstract class and the applicable criteria from their assigned subclasses. Search changes unit implementations and dependencies under a fixed hierarchy; assignment rules and fallback cases appear in Appendix A.

**Operator Composition** Each execution unit performs its task through an operator composition, with prompts and arguments specifying each operator's behavior. The optimizer binds operator inputs to available workflow state or outputs of preceding operators within the unit. Each binding must match the argument type, and the unit's final output must satisfy the inherited I/O contract. Units register selected results as outputs; dependencies on units that consume these outputs form the workflow DAG.

### 4.2 GLOBAL AND LOCAL EVALUATION

**Global Rubric** The global rubric $R_G$ evaluates completed surveys for outline structure, citation accuracy, completeness, content quality, truthfulness, and reference coverage. The resulting score $g(x, y; R_G)$ is averaged over validation topics to obtain $G(W)$ in Equation 1 for candidate comparison.

**Local Rubric** For an execution unit $v_i$, local evaluation combines the shared criteria of $A_i$ with the applicable criteria of subclasses in $\mathcal{C}_i$, using the unit's input as context. Criterion-level scores and evaluation reasons identify unmet requirements and guide the choice of revision targets. Detailed criteria, subclass applicability, and aggregation rules are provided in Appendix H.

### 4.3 WORKFLOW OPTIMIZATION

Let $r$ denote the round and $t$ the search step within a round; each search node represents a complete workflow candidate. At $t = 1$, we initialize the root workflow; at $t \geq 2$, we expand a parent $W^p_{r,t}$ selected from the archive into a child $W^c_{r,t}$ through an edit.

**Initialization** The first round starts from an empty graph and builds an initial workflow by repeatedly selecting abstract classes and composing operators under their contracts. Each composition forms an execution unit, and construction ends at the final document assembly stage.

$$\begin{aligned} W_{r,1} &= \mathrm{Initialize}(\mathcal{H}, \mathcal{O}_r, F_{r-1}), \\ \mathcal{O}_r &= \begin{cases} \mathcal{O}, & r = 1, \\ \mathrm{Ops}(W^*_{r-1}), & r > 1. \end{cases} \end{aligned} \tag{3}$$

Here, $\mathcal{H}$ is the predefined class hierarchy, $\mathcal{O}$ is the full operator catalog, and $F_{r-1}$ is the previous round's review feedback. We set $F_0 = \emptyset$ for the first round. $\mathrm{Ops}(W^*_{r-1})$ denotes the operators used by the previous round's winner. The restriction applies only to initialization; subsequent edits use the full operator catalog. The initial workflow is evaluated and added to the round's archive.

**Selection** Following AFlow's workflow search (Zhang et al., 2025b), we select a parent $W^p_{r,t} = \mathrm{Select}(\mathcal{B}_{r,t-1})$ for $t \geq 2$. $\mathcal{B}_{r,t-1}$ contains candidates evaluated in the round through the previous step, along with their scores and evaluation records. The selected parent's local evaluation results and edit history inform diagnosis.

**Diagnosis and Edit** Local rubric results identify unmet requirements, while review feedback and memory inform proposed implementation or dependency edits. Each node expansion applies one edit to the selected parent workflow. Implementation edits add, remove, or replace operators within a unit, or adjust prompts and arguments. Dependency edits change dataflow between units.

$$\begin{aligned} d_{r,t} &= \mathrm{Diagnose}(W^p_{r,t}, \mathbf{L}^p_{r,t}, F_{r-1}), \\ e_{r,t} &= \mathrm{ProposeEdit}(W^p_{r,t}, d_{r,t}, \mathcal{M}_{r,t-1}), \\ W^c_{r,t} &= \mathrm{Apply}(W^p_{r,t}, e_{r,t}). \end{aligned} \tag{4}$$

$\mathbf{L}^p_{r,t}$ contains the parent's criterion-level local scores and rationales, and $d_{r,t}$ identifies the edit targets and related weaknesses. $\mathcal{M}_{r,t-1}$ stores experience from previous edits. $e_{r,t}$ is the proposed edit applied after passing contract and feasibility checks.

**Evaluation** We compare parent and child global scores on shared validation topics and compute local score changes over comparable criteria for the edit targets.

$$\begin{aligned} \Delta G_{r,t} &= \frac{1}{|D_{r,t}|} \sum_{x \in D_{r,t}} \left[ g(x, W^c_{r,t}(x); R_G) - g(x, W^p_{r,t}(x); R_G) \right], \\ \Delta L_{r,t} &= L_{\mathcal{T}_{r,t}}(W^c_{r,t}) - L_{\mathcal{T}_{r,t}}(W^p_{r,t}). \end{aligned} \tag{5}$$

$D_{r,t}$ contains topics with evaluations for both parent and child; $\Delta G_{r,t}$ is recorded as undefined if no topics are shared. $\mathcal{T}_{r,t}$ specifies the target-unit correspondence before and after the edit and the comparable local criteria. $L_{\mathcal{T}_{r,t}}$ averages criterion scores over target units, then over comparable criteria. $\Delta L_{r,t}$ is recorded as undefined if subclass changes invalidate local comparison or no criteria are comparable. The two deltas indicate changes in overall survey quality and targeted intermediate outputs, respectively.

**Experience Update** We add the evaluated child to the archive and update memory as $\mathcal{M}_{r,t} = \mathrm{Update}(\mathcal{M}_{r,t-1}, d_{r,t}, e_{r,t}, \Delta L_{r,t}, \Delta G_{r,t})$, recording the diagnosis, applied edit, and evaluation results. Edit outcomes accumulate across rounds by subclass and edit type to guide later edits for similar weaknesses. Undefined local deltas are stored separately from zero.

**Round Review and Selection** At the end of each round, we select the workflow with the highest global validation score from the archive and review its surveys.

$$W_r^* = \arg\max_{W \in \mathcal{B}_r} G(W), \qquad F_r = \mathrm{Review}\big(\{W_r^*(x) : x \in D\}\big). \tag{6}$$

$\mathcal{B}_r$ is the candidate set eligible for winner selection at the end of the round. The review summarizes weaknesses in the validation surveys as $F_r$. The winner's operators and review feedback guide initialization in the next round, and the feedback also informs subsequent diagnosis. Among the winners of all $R$ rounds, we select the workflow with the highest $G(W_r^*)$ as the final $W^*$.

### 4.4 Inference Refinement

We edit intermediate outputs during execution of $W^*$ and apply review-based revision to the generated draft (Madaan et al., 2023). The final optimization memory $\mathcal{M}^*$ supports runtime proxy evaluation and alternative implementation selection, while round feedback $\{F_r\}_{r=1}^R$ is consolidated into a review checklist.

**Runtime Editing** For output $z_i = v_i(u_i)$, memory $\mathcal{M}^*$ determines active criteria $\mathcal{Q}_i$ and thresholds $\tau_{i,q}$. Criterion-level proxy scores $\ell_{i,q}$ identify the unmet criteria $\mathcal{F}_i = \{q \in \mathcal{Q}_i : \ell_{i,q}(z_i; u_i) < \tau_{i,q}\}$. If $\mathcal{F}_i \neq \emptyset$, memory guides candidate implementations evaluated on the same input $u_i$ within budget. After I/O and outline preservation checks, a candidate output $z_i'$ is accepted if its mean valid proxy score $\widehat{L}_i$ improves without regression on comparable criteria:

$$\begin{aligned} \Delta\widehat{L}_i &= \widehat{L}_i(z_i'; u_i) - \widehat{L}_i(z_i; u_i), \\ \mathrm{Accept}(z_i') \iff \Delta\widehat{L}_i > \epsilon_{\inf} \quad &\wedge \\ \ell_{i,q}(z_i'; u_i) \geq \ell_{i,q}(z_i; u_i) - \delta, \quad &\forall q \in \mathcal{Q}_i^{\mathrm{cmp}}. \end{aligned} \tag{7}$$

Here, $\mathcal{Q}_i^{\mathrm{cmp}}$ contains comparable criteria, $\epsilon_{\inf}$ is the minimum improvement threshold, and $\delta$ is a numerical tolerance. We pass the first accepted output downstream, retaining $z_i$ if none is accepted.

**Review Revision** We consolidate reviews across rounds into a topic-independent checklist $\mathcal{P} = \mathrm{Distill}(\{F_r\}_{r=1}^R)$. Applying it to the generated draft $y_0$ yields findings $\mathcal{F} = \mathrm{Review}(y_0; \mathcal{P})$. The findings $\mathcal{F}$ guide the selection of sections or subsections and predefined revision sequences, which use retrieved evidence to address the identified weaknesses. Revised outputs pass to final document assembly and citation grounding.

## 5 Experiments

The appendix details the experimental settings (Appendix D), CroftBench construction and evaluation (Appendix C), additional comparisons across retrieval settings, generation models, and topics (Appendix E), and the cost analysis (Appendix G).

### 5.1 Experimental Setup

**Datasets** We evaluate Ozone on SurveyBench (Yan et al., 2025), Bloom-Eval (Zhang et al., 2026a), and our CroftBench. SurveyBench contains 10 computer science (CS) topics with human-written surveys and reference collections. Bloom-Eval includes 10 CS topics and 10 general-domain topics. CroftBench covers 10 CS topics drawn from survey papers published in 2025–2026 and evaluates five components: Content, References, Outline, Figures, and Tables. It extends text-based evaluation to assess how figures and tables support the survey's discussion. Workflow optimization uses a separate collection of CS surveys whose topics do not overlap with any of the three test sets.

**Baselines** We compare Ozone with SurveyForge (Yan et al., 2025), SurveyGen (Bao et al., 2025), and the survey-generation pipeline in the official FlowSearch implementation (Hu et al., 2026). We also include AFlow-SurveyForge, which initializes AFlow (Zhang et al., 2025b) with SurveyForge and optimizes the workflow using the same operator set and global rubric as Ozone.

Table 1: Survey generation results. OZONE-Base omits inference refinement. Bold marks best score.

**(a) SurveyBench**

| Method | Input Cov. | Reference Cov. | **Ref Quality** | **Outline Quality** | Structure | Relevance | Coverage | **Content Quality** | **Quality Average** |
|---|---|---|---|---|---|---|---|---|---|
| SurveyForge | 0.261 | 0.453 | 35.7 | 85.4 | 75.8 | 76.1 | 75.5 | 75.8 | 65.6 |
| SurveyGen | 0.420 | 0.466 | 44.3 | 85.5 | 76.3 | 79.0 | 72.2 | 75.8 | 68.6 |
| FlowSearch | 0.280 | 0.237 | 25.9 | 79.5 | 76.2 | 73.8 | 75.2 | 75.1 | 60.2 |
| AFlow-SurveyForge | 0.258 | 0.408 | 33.3 | 83.8 | 75.5 | 78.5 | 75.5 | 76.5 | 64.5 |
| OZONE-Base | **0.487** | 0.578 | **53.2** | 85.1 | **77.7** | 78.3 | **77.4** | 77.8 | 72.1 |
| OZONE | 0.473 | **0.580** | 52.6 | **85.8** | 77.5 | **79.4** | 77.3 | **78.1** | **72.2** |

**(b) Bloom-Eval: Semantic Scholar search for Paper Retrieval**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | **Average** |
|---|---|---|---|---|---|---|---|
| SurveyForge | 18.47 | 50.02 | 72.34 | 67.23 | 4.59 | **42.26** | 42.49 |
| SurveyGen | 19.31 | **59.14** | 67.28 | **75.59** | **9.23** | 39.14 | 44.95 |
| FlowSearch | 18.79 | 53.90 | 57.46 | 62.93 | 6.41 | 41.04 | 40.09 |
| AFlow-SurveyForge | **19.61** | 55.24 | 71.33 | 68.59 | 7.53 | 39.20 | 43.58 |
| OZONE-Base | 19.52 | 55.32 | **80.76** | 70.62 | 7.38 | 41.67 | 45.88 |
| OZONE | 19.34 | 56.35 | 79.94 | 72.00 | 6.40 | 42.09 | **46.02** |

**(c) CROFTBench**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | Tax. | Tab. Place. | Fig. Place. | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| SurveyForge | 17.95 | 55.13 | 73.03 | 70.83 | 5.01 | **47.64** | 22.21 | 30.0 | 18.0 | 37.76 |
| SurveyGen | 17.18 | **67.07** | 64.86 | **72.61** | 4.81 | 40.78 | 16.41 | 24.0 | 28.0 | 37.30 |
| FlowSearch | 17.56 | 54.14 | 51.30 | 63.37 | 5.28 | 42.95 | **35.00** | 6.0 | 28.0 | 33.73 |
| AFlow-SurveyForge | 19.38 | 62.56 | 72.13 | 70.65 | 4.79 | 43.01 | 19.05 | 52.0 | **42.0** | 42.84 |
| OZONE-Base | 18.97 | 60.06 | 81.24 | 70.97 | 7.12 | 43.37 | 16.17 | 44.0 | 34.0 | 41.77 |
| OZONE | **19.55** | 59.27 | **81.27** | 71.15 | **7.13** | 44.14 | 17.53 | **56.0** | 34.0 | **43.34** |

**Implementation Details** We use GPT-4o-mini for optimization, execution, and rubric evaluation, and Claude Haiku 4.5 for round reviews. We select the workflow with the highest global score on the optimization set within a budget of 3 rounds and 10 iterations per round. All methods use GPT-4o-mini with the SurveyForge retrieval database on SurveyBench and Semantic Scholar search on Bloom-Eval and CROFTBench. SurveyBench outline and content scores are averaged across GPT-4o and Claude Sonnet 4.5, while Bloom-Eval follows its official evaluation settings with GPT-5-mini.

**Metrics** SurveyBench evaluates reference quality through Input Coverage and Reference Coverage, outline quality, and content quality through Structure, Relevance, and Coverage. Quality Average equally weights reference, outline, and content quality. Bloom-Eval groups 16 metrics into six dimensions: Memory (Mem.), Comprehension (Comp.), Application (Appl.), Analysis (Anal.), Evaluation (Eval.), and Creation (Creat.). We report dimension-wise means and their equally weighted average. CROFTBench combines the Bloom-Eval text metrics with taxonomy evaluation (Tax.), table placement (Tab. Place.), and figure placement (Fig. Place.).

### 5.2 EXPERIMENTAL RESULTS AND ANALYSIS

**Main Results** Table 1 shows that OZONE with inference refinement achieves the highest average scores across all three benchmarks under the reported GPT-4o-mini settings. Figure 3 shows that OZONE scores highest on all 10 SurveyBench topics and on 11 of 20 Bloom-Eval topics against all baselines. These results indicate that workflow search guided by shared writing requirements and task-specific specialization improves survey quality.

**Writing Requirements as a Basis for Workflow Search** The fixed OZONE workflow outperforms all baselines in average score on all three benchmarks, supporting its reuse on unseen topics. On SurveyBench, AFlow-SurveyForge scores below its starting workflow, SurveyForge, despite searching both implementations and dependencies. This observation motivates richer search feedback than document-level scores alone. OZONE combines a class hierarchy, local rubric diagnosis,

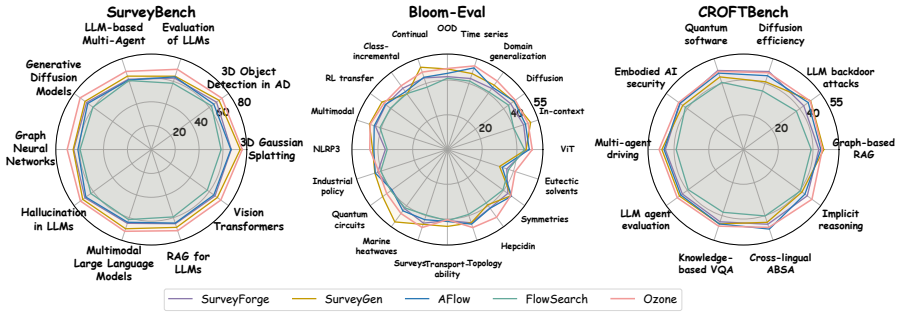



Figure 3: Topic-wise performance with GPT-4o-mini on three benchmarks.

and edit memory, and constructs its initial workflow from an empty graph. Gains in reference coverage and content quality on SurveyBench, together with stronger Application scores on Bloom-Eval and CROFTBench, align with the goal of connecting task-level requirements to document quality.

**The Role of Inference Refinement** Table 1 also reports OZONE-Base, which omits inference refinement. OZONE-Base outperforms all baselines on SurveyBench and Bloom-Eval, and inference refinement adds further gains, most notably on CROFTBench. These results favor reusing the optimized workflow, with inference refinement offering modest improvements.

**Rubric Ablation** Figure 4 compares search with the global rubric alone and with both global and local rubrics. We restrict the comparison to the first round, where no review feedback exists ($F_0 = \emptyset$) and both settings share the initial workflow, 10 validation topics, and a budget of 10 iterations; later rounds start from each setting's own winner. The global-only setting retains edit memory while the global–local setting does not, so the comparison is conservative for the local rubric. Both settings improve the global score in 2 of 9 edits, but local diagnosis yields gains of 0.013 and 0.025, compared with 0.007 and 0.003 without it. From a shared score of 0.662, the global–local setting reaches 0.678 and the global-only setting 0.671, both above SurveyGen at 0.605 and SurveyForge at 0.577 under the same rubric and judge. Local diagnosis also produces larger declines, with a median change of $-0.011$ compared with $-0.004$, suggesting larger revisions of the workflow. About a third of per-topic changes exceed 0.05 in magnitude, so we treat this single-run comparison as a diagnostic of search behavior rather than a significance test.

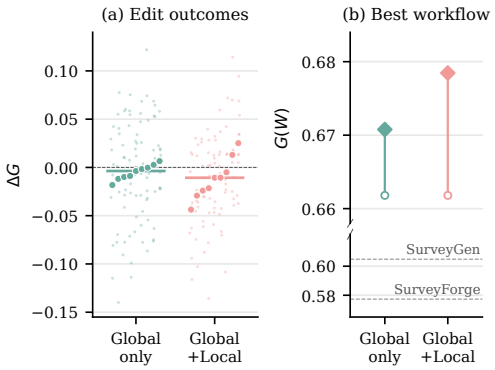



Figure 4: First-round search with and without local rubrics, scored by the global rubric on 10 validation topics. (a) Solid dots: topic-mean $\Delta G$ per edit; faint dots: per-topic changes; bars: medians. (b) Best $G(W)$ from the shared initial workflow (open circles).

**Human Evaluation** We examine whether gains under LLM judges align with human judgments. Six experts in NLP and machine learning score anonymized surveys from OZONE, OZONE-Base and SurveyForge on all 40 test topics (Appendix F). Figure 5 shows that both configurations win more often than they lose against SurveyForge on every criterion. The margins are largest on Structure and Relevance, where OZONE-Base wins 46.2% and 47.9% of comparisons and loses 15.0% and 16.7%. These gains are significant for both configurations ($p < 0.001$, Wilcoxon signed-rank test), and all six annotators favor OZONE on Structure, as do five of six on Relevance. Hierarchy and Coherence show no significant difference, consistent with the comparable Outline Quality on SurveyBench. Chance-corrected agreement is low because scores concentrate between 3 and 5, as is common for Likert ratings of long texts (Amidei et al., 2019; Karpinska et al., 2021), and label order is fixed within each benchmark. We therefore treat these results as support for the direction of the automatic results rather than as a precise estimate of human preference.

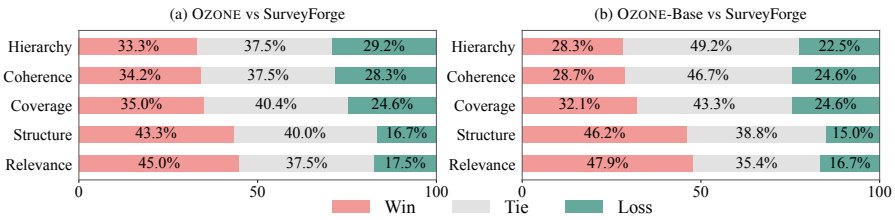



Figure 5: Human preferences of OZONE and OZONE-Base against SurveyForge, pooled across benchmarks. Win and loss refer to the OZONE configuration in each panel. Hierarchy assesses section–subsection relationships; Coherence assesses topic progression and section ordering.

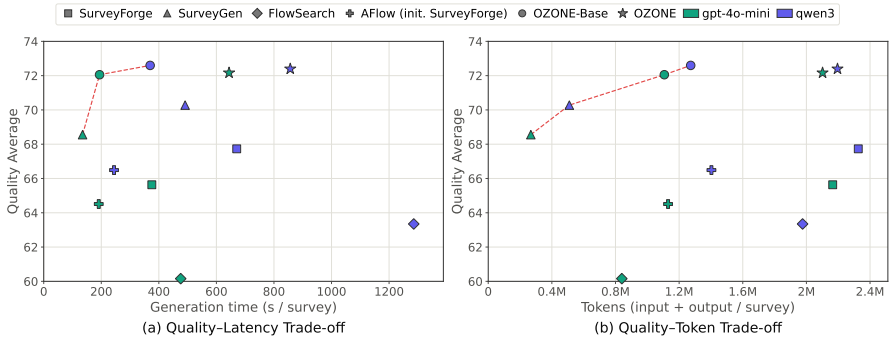



Figure 6: Quality–cost trade-offs on SurveyBench with GPT-4o-mini and Qwen3 (Appendix E.2); the dashed line marks the Pareto frontier.

**Cost Analysis** Figure 6 relates SurveyBench quality to generation time and token usage per survey. OZONE-Base lies on the Pareto frontier for both generation models. Its gains do not come from additional computation; with GPT-4o-mini, it uses time and tokens comparable to AFlow-SurveyForge but scores 7.6 points higher. Although the search objective has no cost term, it also uses about half the tokens of SurveyForge. Inference refinement more than doubles generation time with little change in quality, so OZONE-Base offers the better trade-off.

## 6 CONCLUSION

We propose OZONE (Object-Oriented Optimization of Agentic Workflows) for long-form scientific survey generation. We represent shared writing requirements and task-specific specialization through a predefined class hierarchy, and optimize operator implementations and their DAG composition under inherited contracts. Survey-specific local and global rubrics guide workflow edits by connecting intermediate-output diagnosis with evaluation of completed surveys. Edit histories and round-level reviews inform subsequent search, while runtime editing and review-based refinement support inference-time adaptation. With GPT-4o-mini and inference refinement, OZONE outperforms SurveyGen, a manually designed workflow, by 3.6 points on SurveyBench, 1.07 on BloomEval, and 6.04 on CROFTBench. We introduce CROFTBench to evaluate recent survey topics and extend assessment to figures and tables. By incorporating survey-writing requirements into search representation and evaluation, OZONE improves both generated documents and the procedures used to produce them.

**Limitations and Future Work** OZONE relies on a predefined class hierarchy and writing rubrics, and new domains may require extending them. Search and most evaluations use LLM judges, which may not fully reflect human preferences. Future work will learn task types and evaluation criteria from execution experience and account for execution cost during search. We also plan to evaluate figure and table placement during search and to expand CROFTBench to more topics and disciplines.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with translation, improve the readability of the manuscript, and search for related work. We reviewed the AI-assisted material and take responsibility for the final content of this paper.

## ETHICS STATEMENT

Our human evaluation involved six expert volunteers who were informed of the study purpose; no personally identifiable information was collected (Appendix F). All benchmarks and retrieval corpora consist of publicly available scientific papers and surveys. Generated surveys may contain unsupported claims or incorrect citations, so they should assist rather than replace expert authors, and their content should be verified before use.

## REPRODUCIBILITY STATEMENT

Section D of the appendix specifies datasets, evaluation splits, retrieval settings, and search budgets, and Appendix D.3 lists the model identifier used for each role. Appendix B gives pseudocode for workflow optimization and inference, Appendix H describes the global and local rubrics, and Appendix I provides the prompts. Appendix C details the construction and evaluation of CROFTBench, Appendix F the human evaluation protocol, and Appendix G the optimization and inference costs. SurveyBench results average 10 generations per topic, and all LLM judges are accessed through public APIs with the model versions listed in Appendix D.3.

## REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.19457.

Jacopo Amidei, Paul Piwek, and Alistair Willis. Agreement is overrated: A plea for correlation to assess human evaluation reliability. In *Proceedings of the 12th International Conference on Natural Language Generation*, pp. 344–354, Tokyo, Japan, 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8642. URL https://aclanthology.org/W19-8642/.

Tong Bao, Mir Tafseer Nayeem, Davood Rafiei, and Chengzhi Zhang. SurveyGen: Quality-aware scientific survey generation with large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2712–2736. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.136. URL https://aclanthology.org/2025.emnlp-main.136/.

Yu Chao, Siyu Lin, Xiaorong Wang, Zhu Zhang, Zihan Zhou, Haoyu Wang, Shuo Wang, Jie Zhou, Zhiyuan Liu, and Maosong Sun. LLM×MapReduce-V3: Enabling interactive in-depth survey generation through a MCP-driven hierarchically modular agent system. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 688–695, 2025. doi: 10.18653/v1/2025.emnlp-demos.51. URL https://aclanthology.org/2025.emnlp-demos.51/.

Jing Chen, Zhiheng Yang, Yixian Shen, Jie Liu, Adam Belloum, Paola Grosso, and Chrysa Papagianni. SurveyGen-I: Consistent scientific survey generation with evolving plans and memory-guided writing. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3687–3714, 2025. URL https://aclanthology.org/2025.ijcnlp-long.193/.

Shengda Fan, Xin Cong, Zhong Zhang, Yuepeng Fu, Yesai Wu, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. Generalizing experience for language agents with hierarchical MetaFlows. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2149. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5c882988ce5fac487974ee4f415b96a9-Abstract-Conference.html.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2024.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6465–6488, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.398. URL https://aclanthology.org/2023.emnlp-main.398/.

Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=xYlJRpzZtsY.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/cfd7bee7a651ee9af525098ef67a9e45-Abstract-Conference.html.

Beichen Guo, Zhiyuan Wen, Yu Yang, Peng Gao, Ruosong Yang, and Jiaxing Shen. Sgsimeval: A comprehensive multifaceted and similarity-enhanced benchmark for automatic survey generation systems, 2025. URL https://arxiv.org/abs/2508.11310.

Beichen Guo, Zhiyuan Wen, Jia Gu, Haochen Shi, Jian Wang, Senzhang Wang, Haoyang Li, Ruosong Yang, and Shuaiqi Liu. Surveylens: A discipline-aware benchmark for automatic survey generation, 2026. URL https://arxiv.org/abs/2602.11238.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Spotting llms with binoculars: Zero-shot detection of machine-generated text, 2024.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Yusong Hu, Runmin Ma, Yue Fan, Jinxin Shi, Zongsheng Cao, Yuhao Zhou, Jiakang Yuan, Shuaiyu Zhang, Shiyang Feng, Xiangchao Yan, Shufei Zhang, Wenlong Zhang, Lei Bai, and Bo Zhang. FlowSearch: Advancing deep research with dynamic structured knowledge flow. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21212–21245, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.971. URL https://aclanthology.org/2026.acl-long.971/.

Alon Jacovi, Andrew Wang, Chris Alberti, Connie Tao, Jon Lipovetz, Kate Olszewska, Lukas Haas, Michelle Liu, Nate Keating, Adam Bloniarz, Carl Saroufim, Corey Fry, Dror Marcus, Doron Kukliansky, Gaurav Singh Tomar, James Swirhun, Jinwei Xing, Lily Wang, Madhu Gurumurthy, Michael Aaron, Moran Ambar, Rachana Fellinger, Rui Wang, Zizhao Zhang, Sasha Goldshtein, and Dipanjan Das. The facts grounding leaderboard: Benchmarking llms' ability to ground responses to long-form input, 2025. URL https://arxiv.org/abs/2501.03200.

Marzena Karpinska, Nader Akoury, and Mohit Iyyer. The perils of using Mechanical Turk to evaluate open-ended text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1265–1285, Online and Punta Cana, Dominican Republic,

2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.97. URL https://aclanthology.org/2021.emnlp-main.97/.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Avishek Lahiri, Yufang Hou, and Debarshi Kumar Sanyal. TaxoAlign: Scholarly taxonomy generation using language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 30203–30223, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1536. URL https://aclanthology.org/2025.emnlp-main.1536/.

Tian Lan, Wenwei Zhang, Chen Xu, Heyan Huang, Dahua Lin, Kai Chen, and Xian-Ling Mao. Criticeval: Evaluating large-scale language model as critic. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ZsxZ65YqL1.

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. CriticBench: Benchmarking LLMs for critique-correct reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1552–1587, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.91. URL https://aclanthology.org/2024.findings-acl.91/.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Yuxin Liu, Jinxuan Zhang, Yuezhang Peng, Hefeng Zhou, Xiangfeng Wang, Jiong Lou, Chentao Wu, Jie Li, Jingjing Qu, and Chaochao Lu. Evolving agentic workflow driven by human-agent collaboration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24960–24969, 2026. doi: 10.18653/v1/2026.findings-acl.1250. URL https://aclanthology.org/2026.findings-acl.1250/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. ReCEval: Evaluating reasoning chains via correctness and informativeness. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10066–10086, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.622. URL https://aclanthology.org/2023.emnlp-main.622/.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.06153.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. Assisting in writing Wikipedia-like articles from scratch with large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6252–6278, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.347. URL https://aclanthology.org/2024.naacl-long.347/.

Hwanjun Song, Hang Su, Igor Shalyminov, Jason Cai, and Saab Mansour. FineSurE: Fine-grained summarization evaluation using LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 906–922, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.51. URL https://aclanthology.org/2024.acl-long.51/.

Weihang Su, Anzhe Xie, Qingyao Ai, Jianming Long, Xuanyi Chen, Jiaxin Mao, Ziyi Ye, and Yiqun Liu. Surge: A benchmark and evaluation framework for scientific survey generation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, pp. 3404–3412, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725999. doi: 10.1145/3805712.3808598. URL https://doi.org/10.1145/3805712.3808598.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, Wei Ye, Shikun Zhang, and Yue Zhang. AutoSurvey: Large language models can automatically write surveys. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d07a9fc7da2e2ec0574c38d5f504d105-Abstract-Conference.html.

Shirley Wu, Parth Sarthi, Shiyu Zhao, Aaron Lee, Herumb Shandilya, Adrian Mladenic Grobelnik, Nurendra Choudhary, Eddie Huang, Karthik Subbian, Linjun Zhang, Diyi Yang, James Zou, and Jure Leskovec. Optimas: Optimizing compound AI systems with globally aligned local rewards. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.03041.

Xiangchao Yan, Shiyang Feng, Jiakang Yuan, Renqiu Xia, Bin Wang, Lei Bai, and Bo Zhang. SURVEYFORGE: On the outline heuristics, memory-driven generation, and multi-dimensional evaluation for automated survey writing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12444–12465, 2025. doi: 10.18653/v1/2025.acl-long.609. URL https://aclanthology.org/2025.acl-long.609/.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. FLASK: Fine-grained language model evaluation based on alignment skill sets. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/f41b4a6b202adcd8e150a9d4f124d8f6-Abstract-Conference.html.

Fei Zhang, Zhe Zhao, HaiBin Wen, Tianshuo Wei, Zaixi Zhang, Chao Yang, and Ye Wei. BloomEval: A hierarchical evaluation benchmark for automatic survey generation based on bloom's taxonomy. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28512–28544, 2026a. doi: 10.18653/v1/2026.acl-long.1315. URL https://aclanthology.org/2026.acl-long.1315/.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 75834–75852, 2025a. URL https://proceedings.mlr.press/v267/zhang25bi.html.

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*, 2025b.

URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2510.04618.

Mingming Zhao, Xiaokang Wei, Yuanqi Shao, Kaiwen Zhou, Lin Yang, Siwei Rao, Junhui Zhan, and Zhitang Chen. $A^2$Flow: Automating agentic workflow generation via self-adaptive abstraction operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. URL https://arxiv.org/abs/2511.20693.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Arik. Multi-agent design: Optimizing agents with better prompts and topologies. In *International Conference on Learning Representations*, volume 2026, pp. 15844–15872, 2026.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80569–80611, 2025. URL https://proceedings.mlr.press/v267/zhuge25a.html.

# APPENDIX CONTENTS

**A Class Hierarchy and Subclass Implementations** ..... 15
A.1 Abstract Classes and Contracts ..... 15
A.2 Predefined Subclass Space ..... 16
A.3 Assignment and Reclassification ..... 16
A.4 Rubric Applicability and Mapping Boundaries ..... 17

**B Workflow Optimization and Inference Algorithms** ..... 17

**C CROFTBench** ..... 19
C.1 Collection Scope ..... 19
C.2 Reference Survey Selection ..... 19
C.3 Evaluation Topics and Reference Surveys ..... 20
C.4 Generation and Evaluation ..... 20
C.5 Evaluation Metrics ..... 20

**D Experimental Setup** ..... 22
D.1 Datasets and Evaluation Splits ..... 22
D.2 Implementation Details ..... 22
D.3 Model Card ..... 22
D.4 Workflow Optimization ..... 23
D.5 Evaluation Protocol ..... 23

**E Additional Experiments** ..... 23
E.1 Bloom-Eval across Retrieval Settings ..... 23
E.2 Results with Qwen3 ..... 25

**G Computational Cost and Latency** ..... 30
G.1 Cost Accounting ..... 30
G.2 Latency and Throughput ..... 30
G.3 Reporting Tables ..... 30

**H Global and Local Rubrics** ..... 31
H.1 Evaluation Scope ..... 31
H.2 Global Rubric ..... 31
H.3 Local Rubric Inventory and Applicability ..... 33
H.4 Local Scoring and Diagnosis ..... 36
H.5 Edit Assessment ..... 36

**I Prompt Templates** ..... 36
I.1 Workflow Construction ..... 36
I.2 Local Diagnosis and Edit Proposal ..... 37
I.3 Round Review and Feedback ..... 38
I.4 Inference-Time Review and Refinement ..... 41

# A CLASS HIERARCHY AND SUBCLASS IMPLEMENTATIONS

## A.1 ABSTRACT CLASSES AND CONTRACTS

OZONE organizes the survey generation process into five abstract classes: Retrieve, Analysis, Writing, Verify, and Refine. Each abstract class's I/O contract defines the permitted set of input types, together with the output type and output binding. Each execution unit's input must satisfy one of the input type sets permitted by its contract. The optimizer binds operator inputs to available workflow state or the outputs of preceding operators, forming an executable operator composition. An inherited contract means that a subclass retains the constraints of its parent abstract class, and this relationship is expressed through the class hierarchy and the subclass assignment of each unit.

Table 2 shows the role and allowed output types of each abstract class. Each execution unit belongs to one abstract class that constrains its outputs to later units. PaperPool is the candidate set of papers for a topic, and PaperSet holds the paper records that writers read and cite. SectionPapers keeps a separate PaperSet for each section, and Materials returns a PaperSet together with SurveyExamples. OutlineMD contains the outline and its parsed structure. ContentGrid organizes the body text by section and subsection and is accompanied by the identifiers of the papers supplied to the writer. Final document assembly is a fixed stage outside the five abstract classes, and search does not modify it.

Table 2: Roles and output types declared by the abstract class contracts. Some output types have no corresponding predefined subclass.

| Class | Purpose | Allowed output types |
|---|---|---|
| Retrieve | Retrieve papers and survey examples | PaperPool, PaperSet, SectionPapers, SurveyExamples, Materials |
| Analysis | Analyze literature and define scope and queries | ScopeSpec, AnalysisNotes, Queries |
| Writing | Write outlines, body text, and framing sections | OutlineMD, ContentGrid, FrameText |
| Verify | Check outlines, content, citations, and coverage | Findings, ContentGrid |
| Refine | Revise outlines and body text | OutlineMD, ContentGrid |

Table 3: Declared output types and operators for each predefined subclass. Units are assigned one or more subclasses based on the producing operators of registered outputs and the operator composition within each unit.

| Subclass | Output type | Declared operators |
|---|---|---|
| Pool retrieval | PaperPool | Paper pool retrieval |
| Paper record retrieval | PaperSet | Paper metadata retrieval |
| Scoped retrieval | PaperSet, SectionPapers | Section retrieval, Subsection retrieval |
| Example retrieval | SurveyExamples | Survey example retrieval |
| External retrieval | PaperSet | External paper retrieval |
| Literature analysis | AnalysisNotes | Paper evidence notes, Topic analysis |
| Comparative analysis | AnalysisNotes | Comparative synthesis |
| Scope analysis | ScopeSpec, Queries | Query expansion |
| Gap analysis | Queries | Coverage gap analysis, Gap-directed query generation |
| Outline writing | OutlineMD | Initial outline generation, Outline integration, Subsection outline generation, Outline compaction |
| Content writing | ContentGrid | Subsection writing |
| Frame writing | FrameText | Frame writing |
| Outline verification | Findings | Outline critique |
| Content verification | Findings | Content critique |
| Citation verification | Findings | Citation verification |
| Coverage verification | Findings | Coverage gap analysis |
| Outline refinement | OutlineMD | Outline deduplication, Outline compaction |
| Content refinement | ContentGrid | Content revision |
| Coherence refinement | ContentGrid | Local coherence refinement, Global coherence refinement |
| Synthesis refinement | ContentGrid | Comparative synthesis |

### A.2 Predefined Subclass Space

The predefined class hierarchy defines 20 subclasses: five under Retrieve, four under Analysis, three under Writing, four under Verify, and four under Refine. These subclasses are fixed before search, and search builds and modifies the operator implementation of each unit. A subclass can correspond to several implementations and execution units, and a unit can correspond to several subclasses of the same parent abstract class. Table 3 lists the output types and operators declared for each subclass. The declared operators are used only to identify subclasses and do not restrict which operators a unit can use. A unit is assigned its subclasses after its implementation is built, based on its outputs and operator composition.

### A.3 Assignment and Reclassification

Each subclass in Table 3 represents a task and declares its output type and the operators used to identify it. We assign subclasses to execution units after initial composition and reassign them after every edit. These assignments determine which local rubric questions apply to each unit. For unit $v_i$,

the subclass set $\mathcal{C}_i$ depends on its parent abstract class $A_i$, operator composition $O_i$, and registered outputs and bindings $B_i$. Each binding in $B_i$ identifies the operator that produces an output passed to a downstream unit. We consider only subclasses declared under $A_i$ and assign them in the following order:

1. If the producing operator and output type are declared by the same subclass, assign that subclass.
2. If the producing operator does not identify a subclass, consider subclasses that declare the output type and select the one with the most declared operators in $O_i$.
3. If no subclass declares the output type but the producing operator is identified, select the subclass that declares that operator.
4. Add subclasses whose declared operators and output types match the unit's composition and registered outputs. Remove duplicates to form $\mathcal{C}_i$.

For example, a unit that registers PaperPool does not receive Paper record retrieval merely because it retrieves paper metadata internally. That subclass requires a registered PaperSet output. A unit that performs both comparative synthesis and content revision and registers ContentGrid receives both Synthesis refinement and Content refinement.

We assign subclass sets after initial composition. After each edit, we reclassify units based on their final operators and registered outputs, then record changes by comparing their subclass sets with those of units with the same declared operators in the parent workflow. Execution units form a DAG through dependencies between compatible I/O interfaces. We distinguish these dataflow dependencies from inheritance in the class hierarchy. When replacing an implementation, we check the argument types of its input bindings against the inherited I/O contract.

### A.4 Rubric Applicability and Mapping Boundaries

The local rubric contains questions shared by a parent abstract class and questions specific to its subclasses. We denote the subclasses to which question $q$ applies by $\mathcal{A}(q)$ and define the applicable question set for a classified unit as follows:

$$\mathcal{Q}_i = \{q \in \mathcal{Q}(A_i) \mid \mathcal{A}(q) = \emptyset \ \vee \ \mathcal{A}(q) \cap \mathcal{C}_i \neq \emptyset\}. \tag{8}$$

$\mathcal{Q}(A_i)$ contains all questions for the parent abstract class, and $\mathcal{A}(q) = \emptyset$ denotes a shared question. A question applicable to multiple subclasses is evaluated only once. If an edit changes the subclass set of the target unit, we invalidate the local score change for that edit. If no subclass corresponds to a declared output, we record the unit as unclassified and apply the parent abstract class questions without subclass filtering. If the target unit is reclassified after an edit, its local scores before and after the edit are based on different question sets. We therefore evaluate that edit using only the global score change.

## B Workflow Optimization and Inference Algorithms

Algorithm 1 summarizes the optimization procedure. Each search node is a complete workflow; an expansion edits a parent selected from the archive. The budget $T$ includes root initialization and up to $T - 1$ child expansions per round. Memory $\mathcal{M}$ persists across rounds. Algorithm 1 omits round and step indices on parent and child workflows, evaluation records, and memory.

$\mathrm{Initialize}$ assigns subclasses from operator compositions and output bindings; $\mathrm{Apply}$ updates these assignments after an edit using Equation 2. Archive entries retain scores and evaluation records alongside each workflow. Parent selection follows AFlow (Zhang et al., 2025b); proposed edits pass contract and feasibility checks before application. Local deltas use comparable target criteria, with undefined comparisons recorded separately from zero.

**Inference Procedure** Algorithm 2 applies runtime editing during workflow execution and review revision before final document assembly. The default runtime policy accepts the first candidate that satisfies Equation 7.

**Algorithm 1** Object-Oriented Workflow Optimization

**Require:** Class hierarchy $\mathcal{H}$, operator catalog $\mathcal{O}$, validation topics $D$, global rubric $R_G$, subclass-specific local rubrics, rounds $R$, search budget $T$
**Ensure:** Selected workflow $W^*$, memory $\mathcal{M}^*$, round feedback $\{F_r\}_{r=1}^{R}$

$\mathcal{M} \leftarrow \emptyset$; $F_0 \leftarrow \emptyset$
**for** $r = 1, \ldots, R$ **do**
  $\mathcal{O}_r \leftarrow \mathcal{O}$ if $r = 1$, otherwise $\mathrm{Ops}(W^*_{r-1})$
  $W_{r,1} \leftarrow \mathrm{Initialize}(\mathcal{H}, \mathcal{O}_r, F_{r-1})$
  Execute $W_{r,1}$ on $D$; record outputs, $G(W_{r,1})$, and local evaluations
  $\mathcal{B} \leftarrow \{W_{r,1}\}$
  **for** $t = 2, \ldots, T$ **do**
    $W^p \leftarrow \mathrm{Select}(\mathcal{B})$ (AFlow parent selection)
    Retrieve the parent's local evaluation record $\mathbf{L}^p$
    $d \leftarrow \mathrm{Diagnose}(W^p, \mathbf{L}^p, F_{r-1})$
    $e \leftarrow \mathrm{ProposeEdit}(W^p, d, \mathcal{M})$ using the full catalog $\mathcal{O}$
    $W^c \leftarrow \mathrm{Apply}(W^p, e)$
    Execute $W^c$ on $D$; record outputs, $G(W^c)$, and local evaluations
    Compute $\Delta G$ and $\Delta L$ using Equation 5
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{W^c\}$
    $\mathcal{M} \leftarrow \mathrm{Update}(\mathcal{M}, d, e, \Delta L, \Delta G)$
  **end for**
  $\mathcal{B}_r \leftarrow$ candidates eligible for winner selection in $\mathcal{B}$
  $W^*_r \leftarrow \arg\max_{W \in \mathcal{B}_r} G(W)$
  $F_r \leftarrow \mathrm{Review}(\{W^*_r(x) : x \in D\})$ using recorded surveys
**end for**
$W^* \leftarrow \arg\max_{W \in \{W^*_1, \ldots, W^*_R\}} G(W)$; $\mathcal{M}^* \leftarrow \mathcal{M}$
**return** $W^*, \mathcal{M}^*, \{F_r\}_{r=1}^{R}$

**Algorithm 2** Survey Generation with Inference Refinement

**Require:** Topic $x$, workflow $W^*$, memory $\mathcal{M}^*$, round feedback $\{F_r\}_{r=1}^{R}$, runtime budget
**Ensure:** Final survey $y$

$\mathcal{P} \leftarrow \mathrm{Distill}(\{F_r\}_{r=1}^{R})$
Initialize workflow state $s$ with $x$
**for** each unit $v_i$ in dependency order, before final assembly **do**
  Bind input $u_i$ from $s$; $z_i \leftarrow v_i(u_i)$
  **if** runtime evaluation is enabled for $v_i$ and budget remains **then**
    $(\mathcal{Q}_i, \tau_i) \leftarrow \mathrm{ProxyConfig}(v_i, \mathcal{M}^*)$
    $a_i \leftarrow \mathrm{ProxyEval}(z_i, u_i, \mathcal{Q}_i)$
    $\mathcal{F}_i \leftarrow \{q \in \mathcal{Q}_i : a_i[q] < \tau_{i,q}\}$
    **if** $\mathcal{F}_i \neq \emptyset$ **then**
      $\mathcal{K}_i \leftarrow \mathrm{Candidates}(v_i, \mathcal{F}_i, \mathcal{M}^*)$
      **for** $k \in \mathcal{K}_i$ **do**
        **if** the remaining budget is insufficient for another attempt **then**
          **break**
        **end if**
        Execute candidate $k$ on $u_i$ to obtain $z'_i$
        **if** I/O format and outline preservation checks pass **then**
          $a'_i \leftarrow \mathrm{ProxyEval}(z'_i, u_i, \mathcal{Q}_i)$
          **if** $a'_i$ and $a_i$ satisfy Equation 7 **then**
            $z_i \leftarrow z'_i$; **break**
          **end if**
        **end if**
      **end for**
    **end if**
  **end if**
  Register $z_i$ in $s$ for downstream units
**end for**
Obtain draft $y_0$ from $s$; $\mathcal{F} \leftarrow \mathrm{Review}(y_0; \mathcal{P})$
**for** each section or subsection $j$ selected by finding severity and count **do**
  Select operator sequence $\pi$ for $j$ by finding type and location
  $s \leftarrow \mathrm{Revise}(s, j, \pi, \mathcal{F})$
**end for**
Assemble the document from $s$ and ground citations to obtain $y$
**return** $y$

Table 4: CROFTBench collection and selection stages. The 35 candidates after deduplication comprise 29 papers from 2025 and six from 2026.

| Selection stage | Papers |
|---|---|
| Collected arXiv CS papers | 13,437 |
| Surveys identified from titles and abstracts | 455 |
| Surveys with taxonomy figures and method comparison tables available in LaTeX source | 185 |
| Shortlist based on topics and citation counts | 50 |
| Candidates after deduplication | 35 |
| Final selection with the lowest Binoculars scores | 10 |

**Runtime functions.** $\mathrm{ProxyConfig}$ determines active criteria $\mathcal{Q}_i$ and thresholds $\tau_i$ from criterion scores and edit outcomes in $\mathcal{M}^*$. $\mathrm{ProxyEval}$ evaluates an output in its input context and returns $a_i[q] = \ell_{i,q}(z_i; u_i)$ and the mean $\widehat{L}_i(z_i; u_i)$ over valid active criteria. $\mathrm{Candidates}$ retrieves alternative implementations using unmet criteria and memory; configured argument adjustments and revision operators may also be included. Each candidate runs on the same input $u_i$ and is reevaluated with the same active criteria after passing I/O format and outline preservation checks. Memory guides candidate selection; acceptance uses proxy scores reevaluated on the current input under Equation 7. The minimum improvement threshold $\epsilon_{\mathrm{inf}}$ defaults to zero. By default, the first accepted output is passed to downstream units for the current execution; if no candidate is accepted, the original output $z_i$ is retained. Runtime evaluation and candidate attempts consume the available budget.

**Review functions.** $\mathrm{Distill}$ merges similar review items, retains recurring weaknesses, and ranks them by supporting topics and review items. $\mathrm{Review}$ applies the checklist to the draft and returns findings with severity, location, evidence, and revision requirements. Sections or subsections are prioritized by finding severity and count, and predefined operator sequences are selected by weakness type and location. $\mathrm{Revise}$ retrieves evidence and applies the selected operator sequence to target $j$, updating text, headings, and sources in $s$. These sequences perform comparative synthesis, critique and revision, citation verification, coherence refinement, or heading revision. Quantitative comparisons and limitation analysis require supporting retrieved evidence.

# C CROFTBENCH

CROFTBench is a scientific survey generation benchmark that evaluates five components: Content, References, Outline, Figures, and Tables. Its topics and reference surveys are drawn from recent CS surveys, and it extends text-based evaluation with the evaluation of taxonomy figures and method comparison tables.

## C.1 COLLECTION SCOPE

The collection covers all `cs.*` categories of arXiv from January 1, 2025 to July 30, 2026. We collect 13,437 papers, identify surveys from their titles and abstracts, and obtain 455 surveys. Among these, we select the 185 surveys that contain both a taxonomy figure and a method comparison table and whose LaTeX source provides both.

## C.2 REFERENCE SURVEY SELECTION

We first shortlist 50 surveys by topic and citation count and remove duplicates, leaving 35 candidates. This candidate set consists of 29 papers from 2025 and six from 2026. As an additional criterion for narrowing the final evaluation set, we apply Binoculars (Hans et al., 2024) to the candidates, sort them by score in ascending order, and select the ten with the lowest scores. The topic and reference survey of each selected survey are included in the benchmark. Binoculars produces a detection score for which a lower value indicates LLM-generated text. This criterion therefore restricts the evaluation to surveys that the detector judges to resemble LLM-generated text, and CROFTBench evaluates performance on recent CS surveys that satisfy this selection condition. We use this criterion so that the reference surveys are comparable in style to generated surveys, which keeps the text metrics from rewarding stylistic proximity to human writing rather than content. Table 4 summarizes the number of candidates at each stage.

Table 5: The 10 evaluation topics in CROFTBench and their reference surveys.

| Topic | Reference survey | Venue | Year | Citations | Binoculars |
|---|---|---|---|---|---|
| Vulnerabilities and attacks on embodied AI systems | Towards Robust and Secure Embodied AI: A Survey on Vulnerabilities and Attacks | ACM Computing Surveys | 2026 | 80 | 0.9290 |
| Cross-lingual aspect-based sentiment analysis | Cross-lingual Aspect-Based Sentiment Analysis: A Survey on Tasks, Approaches, and Challenges | Information Fusion | 2025 | 25 | 0.9494 |
| Quantum software engineering and quantum computing for software engineering research | Quantum Software Engineering and Potential of Quantum Computing in Software Engineering Research: A Review | Automated Software Engineering | 2025 | 28 | 0.9518 |
| Graph-based retrieval-augmented generation for domain-specialized large language models | A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models | arXiv preprint | 2025 | 321 | 0.9864 |
| Evaluation methods and benchmarks for LLM-based agents | A Survey on Evaluation of LLM-based Agents | Findings of ACL | 2026 | 336 | 0.9942 |
| LLM-based multi-agent autonomous driving systems | Multi-Agent Autonomous Driving Systems with Large Language Models: A Survey of Recent Advances, Resources, and Future Directions | Findings of EMNLP | 2025 | 54 | 1.0000 |
| Efficiency techniques for diffusion models | Efficient Diffusion Models: A Survey | TMLR | 2025 | 81 | 1.0108 |
| Knowledge-based visual question answering systems | A Comprehensive Survey of Knowledge-Based Visual Question Answering Systems: The Lifecycle of Knowledge in Visual Reasoning Task | IEEE TKDE | 2026 | 10 | 1.0204 |
| Implicit reasoning in large language models | Implicit Reasoning in Large Language Models: A Comprehensive Survey | arXiv preprint | 2025 | 47 | 1.0218 |
| Backdoor attacks and defenses in large language models | A Survey on Backdoor Threats in Large Language Models (LLMs): Attacks, Defenses, and Evaluations | Trans. on Artificial Intelligence | 2025 | 97 | 1.0252 |

### C.3 Evaluation Topics and Reference Surveys

Table 5 lists the final evaluation topics, the metadata of the reference surveys, and their median Binoculars scores.

### C.4 Generation and Evaluation

Depending on the survey structure, the module also generates taxonomy (Lahiri et al., 2025) and timeline diagrams. Each table compares methods cited in its section and extracts the source passages supporting its values. A cell is populated only when the extracted passage appears in the cited paper and supports the corresponding value; otherwise, it is left blank. For figures, the module extracts captioned regions from cited PDFs, prioritizes diagrams that explain methods, selects at most one figure per section, and limits the total number according to this priority. Finally, rule-based checks on the LaTeX output verify that the original text and citations are preserved.

The text score is the equally weighted mean of the six Bloom-Eval dimensions defined in the main text. EFid and TFSim each average their F1 and DS scores. For each method and generation model, we generate one survey for each of the 10 topics; GPT-5-mini serves as the primary evaluator, while GPT-4o evaluates citation faithfulness (CF). CROFTBench additionally evaluates taxonomy coverage and the placement of tables and figures, as defined in Appendix C.5.

### C.5 Evaluation Metrics

Because the same figure and table module is applied to every method, variation in these metrics reflects differences in the content and structure of the generated surveys under a fixed module. A survey that identifies appropriate evidence and develops its sections around coherent, distinguishable content provides the module with a stronger basis for selecting and placing figures and tables. Accordingly, the metrics ask two questions: whether the papers promoted in the generated taxon-

omy are also cited by the reference survey, and whether each inserted table or figure is placed in the section whose content best matches what it presents.

**Notation.** For each of the $T = 10$ topics, $\mathcal{L}_t$ denotes the papers on the leaves of the generated taxonomy figure, $\mathcal{R}_t$ all papers cited by the generated survey, $G_t$ the reference list of the reference survey, and $\mathcal{T}_t$ and $\mathcal{F}_t$ the tables and the figures inserted by the module. We use a fixed evaluation budget of $B = 5$ for both tables and figures, matching the figure limit used by the module.

**Taxonomy coverage (Tax.).** The taxonomy figure lists, on each leaf, papers considered representative of a sub-area. Because their representativeness cannot be assessed from the generated survey alone, we use the reference survey as an external criterion: a promoted paper counts as correct if it is also cited by the reference survey. This corresponds to the Reference Coverage of SurveyBench (Yan et al., 2025) restricted to the papers on the taxonomy:

$$\text{Tax.} = \frac{100}{T} \sum_{t=1}^{T} \frac{|\mathcal{L}_t \cap G_t|}{|\mathcal{L}_t|}. \tag{9}$$

**Placement (Tab. Place., Fig. Place.).** We treat a table or figure as well placed when it appears in a section whose text substantively discusses what it shows. We test this by hiding the section in which the module placed an item and asking whether an independent judge, given only the item and the candidate sections, would return it to the same section. If the judge recovers the section, the text of that section and the content of the item agree; if it does not, the item sits where its content is not discussed, or the same content is spread over several sections. For each inserted table $x$, the candidates $S(x)$ are all sections at the same level, with captions and mention sentences removed from their text.

We use a two-stage ranking procedure to first identify semantically plausible sections efficiently and then resolve the most relevant candidates with a judge. Stage 1 ranks $S(x)$ by the cosine similarity between text-embedding-3-small[1] embeddings of the table content (column headers, row labels, and cells, without the caption) and of each section's text. Stage 2 lets a judge re-rank the top three candidates. After stage 2, every candidate section has a rank. Let $r(x)$ be the rank of the section in which the module actually placed $x$; the placement is correct when $r(x) = 1$, that is, when the judge ranks the module's own section first. Figures follow the same procedure, except that stage 1 embeds the figure's original caption from its source paper, because the caption written by the module is conditioned on the section, and stage 2 shows the figure image together with that caption to the judge.

A plain accuracy over inserted items would reward a survey that receives a single well-placed table, although a survey that cannot accommodate tables is not a better survey. We therefore count correct placements against a fixed budget of $B = 5$ slots per survey:

$$\text{Tab. Place.} = \frac{100}{T \cdot B} \sum_{t=1}^{T} \min\Big(B,\ \big|\{x \in \mathcal{T}_t : r(x) = 1\}\big|\Big), \tag{10}$$

and Fig. Place. is defined analogously over $\mathcal{F}_t$. The $\min$ caps each survey at $B$ slots: a survey with two correctly placed tables fills two, and one with six is still counted as five, so no survey gains from inserting more tables than the budget. Over the $T = 10$ topics a metric thus has $T \cdot B = 50$ slots, and each correct item fills one of them, so each correctly placed item is worth $100/50 = 2$ points and the scores are even integers.

**Results** In Table 1(c), OZONE places tables best (56.0): each section cites several papers whose text is retrievable and discusses them in that section, so the module can build a table there and a judge can return it to its section. FlowSearch leads on taxonomy coverage (35.0 vs. 17.5), but its taxonomies contain substantially far fewer papers than those of OZONE (40 vs. 270 over ten topics), so a few matches move the ratio. AFlow-SurveyForge leads on figure placement (42.0 vs. 34.0), but its one-paragraph subsections are highly distinguishable: the same section-recovery test using a held-out paragraph instead of the figure already scores 44.0. By contrast, OZONE uses three to four paragraphs per subsection, making section recovery more demanding.

[1] `https://developers.openai.com/api/docs/models/text-embedding-3-small`

Table 6: The validation topics used during workflow optimization.

| Topic | Reference survey | Venue | Year | Citations |
|---|---|---|---|---|
| Large Language Models | A Comprehensive Overview of Large Language Models | ACM TIST | 2025 | 2,003 |
| Deep Learning Based Text Classification | Deep Learning Based Text Classification: A Comprehensive Review | ACM Computing Surveys | 2021 | 1,378 |
| Deep Neural Networks and Tabular Data | Deep Neural Networks and Tabular Data: A Survey | IEEE TNNLS | 2024 | 1,349 |
| Safe Control with Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction methods | Safe Control with Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction methods | IEEE T-RO | 2023 | 449 |
| Dense Text Retrieval based on Pretrained Language Models | Dense Text Retrieval based on Pretrained Language Models: A Survey | ACM TOIS | 2024 | 353 |
| Evolution of Image Segmentation using Deep Convolutional Neural Network | Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey | Knowledge-Based Systems | 2020 | 313 |
| Graph Representation Learning Methods | A Survey on Graph Representation Learning Methods | ACM TIST | 2024 | 271 |
| Vision-and-Language Navigation: A Survey of Tasks, Methods, and Future Directions | Vision-and-Language Navigation: A Survey of Tasks, Methods, and Future Directions | ACL | 2022 | 208 |
| Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence | A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence | arXiv preprint | 2020 | 155 |
| Federated Learning for Computationally-Constrained Heterogeneous Devices | Federated Learning for Computationally-Constrained Heterogeneous Devices: A Survey | ACM Computing Surveys | 2023 | 143 |

Table 7: Default settings for workflow optimization and evaluation.

| Setting | Default configuration |
|---|---|
| Optimizer / executor | GPT-4o-mini |
| Optimizer temperature | 0.7 |
| Local / global optimization judge | GPT-4o-mini |
| Round reviewer | Claude Haiku 4.5 |
| Retrieval embedding | Alibaba-NLP/gte-large-en-v1.5 |
| Retrieval corpus | SurveyForge database, Semantic Scholar |
| Retrieval pool size | 1,000 (default); range 500–2,000 |
| Optimization topics | 10 validation topics |
| Search budget | Up to 3 rounds, up to 10 iterations per round |
| Parent selection | Epsilon-greedy, $\epsilon = 0.3$ |
| SurveyBench evaluation panel | GPT-4o-2024-08-06, Claude Sonnet 4.5 |

# D Experimental Setup

## D.1 Datasets and Evaluation Splits

We evaluate survey generation performance on SurveyBench, Bloom-Eval (Yan et al., 2025; Zhang et al., 2026a), and CROFTBench. The validation set comprises survey papers that do not appear in the SurveyBench, Bloom-Eval, or CROFTBench test sets. Table 6 presents the composition of the validation set.

## D.2 Implementation Details

Table 7 reports the settings used for optimization and evaluation. Because retrieval ranges and reranking settings can vary across workflows, we record the arguments of the selected workflow alongside the experimental settings. SurveyBench uses gte-large-en-v1.5 for retrieval. For Bloom-Eval, we keep the remaining settings fixed and report results separately for the SurveyForge database and Semantic Scholar. The optimizer temperature is 0.7, while the executor temperature follows a separate model configuration. The round reviewer uses a default temperature of 0 and a maximum output budget of 4,096 tokens.

## D.3 Model Card

Table 8 summarizes the roles of the models used in this work.

Table 8: Model card by role, distinguishing API model identifiers from retrieval encoder identifiers. Alternative judge panels are described in the accompanying discussion of execution configurations.

| Role | Default model identifier | Inputs and purpose |
|---|---|---|
| Workflow optimizer | `gpt-4o-mini` | Propose compositions and edits from contracts, diagnoses, and search history |
| Operator executor | `gpt-4o-mini` | Generate analyses, outlines, content, and revisions from prompts and evidence |
| Local / global judge | `gpt-4o-mini` | Evaluate intermediate outputs and surveys against rubrics |
| Round reviewer | `anthropic/claude-haiku-4.5` | Identify survey weaknesses and check review items |
| SurveyBench judge panel | `openai/gpt-4o-2024-08-06`, `anthropic/claude-sonnet-4.5` | Evaluate generated outlines and surveys using benchmark criteria |
| Retrieval encoder | `Alibaba-NLP/gte-large-en-v1.5` | Embed queries and paper text for retrieval |

### D.4 Workflow Optimization

The first step of each round constructs an initial workflow based on the class hierarchy and I/O contracts. From the second round onward, initialization is restricted to the operators in the previous round's winning workflow. Subsequent node expansions use the full operator catalog. Each node expansion selects a parent workflow from the archive and modifies an execution-unit implementation or dependency based on local diagnosis to produce a child workflow. The default parent selection policy is $\varepsilon$-greedy. We execute each candidate on the validation topics and perform local and global evaluations. Edit outcomes are recorded in memory, and the review of the round winner's survey provides feedback for the next round.

### D.5 Evaluation Protocol

SurveyBench outline and content quality scores are averaged across evaluations by GPT-4o and Claude Sonnet 4.5. Following the official Bloom-Eval settings, we use GPT-5-mini for LLM-based extraction, semantic matching, and GRADE scoring. Citation faithfulness (CF) is evaluated with GPT-4o. We report EFid and TFSim as the mean of their F1 and DS scores, and compute the Bloom-Eval average across 16 metrics. For CROFTBench, we use gpt-4o-mini to evaluate figures and tables. Rubric scores used for optimization and external benchmark scores serve different evaluation purposes.

## E Additional Experiments

### E.1 Bloom-Eval across Retrieval Settings

We compare the SurveyForge database and Semantic Scholar search on the same 20 Bloom-Eval topics, retaining the generation and evaluation settings for each model.

**GPT-4o-mini** Table 9 reports GPT-4o-mini results with the SurveyForge database; Semantic Scholar results appear in Table 1(b). Across both retrieval settings, OZONE outperforms the baselines in average score with and without inference refinement. OZONE-Base achieves the highest average with the SurveyForge database. With Semantic Scholar search, the configuration combining runtime editing and review achieves the highest average. The optimized workflow retains its advantage when the retrieval source changes, and the additional gains from refinement are small and depend on the setting.

Table 9: Bloom-Eval results with GPT-4o-mini under two retrieval settings, averaged over 20 topics with one generation per topic. Bold indicates column maxima within each setting.

**(a) SurveyForge DB**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | Average |
|---|---|---|---|---|---|---|---|
| SurveyForge | 18.54 | 52.27 | 72.18 | 66.60 | 5.27 | **44.06** | 43.15 |
| SurveyGen | 17.24 | **58.99** | 66.85 | **75.04** | **8.58** | 37.81 | 44.08 |
| FlowSearch | 15.66 | 47.94 | 57.43 | 61.85 | 6.10 | 38.84 | 37.97 |
| AFlow-SurveyForge | 19.86 | 57.12 | 71.56 | 66.12 | 4.83 | 40.70 | 43.37 |
| OZONE-Base | 20.09 | 56.24 | **80.63** | 70.18 | 6.83 | 41.45 | **45.90** |
| OZONE | 20.06 | 56.14 | 80.59 | 70.01 | 7.25 | 38.97 | 45.50 |

**(b) Semantic Scholar search**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | Average |
|---|---|---|---|---|---|---|---|
| SurveyForge | 18.47 | 50.02 | 72.34 | 67.23 | 4.59 | **42.26** | 42.49 |
| SurveyGen | 19.31 | **59.14** | 67.28 | **75.59** | **9.23** | 39.14 | 44.95 |
| FlowSearch | 18.79 | 53.90 | 57.46 | 62.93 | 6.41 | 41.04 | 40.09 |
| AFlow-SurveyForge | **19.61** | 55.24 | 71.33 | 68.59 | 7.53 | 39.20 | 43.58 |
| OZONE-Base | 19.52 | 55.32 | **80.76** | 70.62 | 7.38 | 41.67 | 45.88 |
| OZONE | 19.34 | 56.35 | 79.94 | 72.00 | 6.40 | 42.09 | **46.02** |

**Qwen3** Table 10 reports Qwen3-30B-A3B-Instruct-2507 results with the SurveyForge database; Semantic Scholar results appear in Table 11(b). With Qwen3, OZONE-Base achieves higher average and Application scores than the baselines under both retrieval settings. As with GPT-4o-mini, OZONE-Base achieves the highest average with the SurveyForge database, and the configuration combining runtime editing and review achieves the highest average with Semantic Scholar search.

Table 10: Bloom-Eval results with Qwen3-30B-A3B-Instruct-2507 under two retrieval settings, averaged over 20 topics with one generation per topic. Bold indicates column maxima within each setting.

**(a) SurveyForge DB**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | Average |
|---|---|---|---|---|---|---|---|
| SurveyForge | **19.24** | 53.69 | 76.32 | 70.52 | 5.77 | **57.13** | 47.11 |
| SurveyGen | 18.69 | **60.55** | 62.60 | **72.60** | 5.29 | 52.19 | 45.32 |
| FlowSearch | 15.84 | 55.09 | 64.03 | 71.02 | 5.58 | 53.40 | 44.16 |
| AFlow-SurveyForge | 18.57 | 55.83 | 75.74 | 66.91 | 5.45 | 55.64 | 46.36 |
| OZONE-Base | 17.67 | 57.71 | **84.14** | 67.69 | 5.87 | 54.59 | **47.94** |
| OZONE | 18.66 | 56.87 | 84.02 | 67.67 | 5.61 | 54.33 | 47.86 |

**(b) Semantic Scholar search**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | Average |
|---|---|---|---|---|---|---|---|
| SurveyForge | 17.99 | 54.22 | 76.04 | 70.83 | 4.36 | **56.78** | 46.71 |
| SurveyGen | **19.33** | **61.50** | 64.51 | **71.77** | 4.08 | 51.88 | 45.51 |
| FlowSearch | 16.44 | 57.99 | 66.71 | 70.52 | 3.36 | 51.63 | 44.44 |
| AFlow-SurveyForge | 19.11 | 58.32 | 74.94 | 67.83 | **7.57** | 54.41 | 47.03 |
| OZONE-Base | 17.55 | 59.33 | 84.16 | 68.67 | 5.01 | 55.82 | 48.42 |
| OZONE | 18.99 | 58.78 | 84.12 | 70.62 | 5.30 | 53.67 | **48.58** |

### E.2 RESULTS WITH QWEN3

Table 11 reports survey-generation results with Qwen3-30B-A3B-Instruct-2507 across the three benchmarks. The metric definitions and aggregation follow the main text, and both Bloom-Eval and CROFTBench use Semantic Scholar search. On CROFTBench, OZONE-Base achieves higher average and Application scores than the baselines. OZONE also outperforms the baselines on average, although OZONE-Base records the highest score. The optimized workflow therefore retains its advantage with Qwen3 without inference refinement.

Table 11: Survey generation results with Qwen3. OZONE-Base omits inference refinement. Bold marks column maxima.

**(a) SurveyBench**

| Method | Input Cov. | Reference Cov. | **Ref Quality** | **Outline Quality** | Structure | Relevance | Coverage | **Content Quality** | **Quality Average** |
|---|---|---|---|---|---|---|---|---|---|
| SurveyForge | 0.269 | 0.382 | 32.5 | **88.1** | 80.9 | **84.4** | **82.5** | **82.6** | 67.7 |
| SurveyGen | 0.419 | 0.430 | 42.5 | 87.5 | 79.6 | 83.0 | 80.0 | 80.9 | 70.3 |
| FlowSearch | 0.189 | 0.305 | 24.7 | 85.2 | 79.5 | 81.6 | 79.1 | 80.1 | 63.3 |
| AFlow-SurveyForge | 0.255 | 0.343 | 29.9 | 87.5 | **81.9** | 82.3 | 82.0 | 82.1 | 66.5 |
| OZONE-Base | **0.468** | **0.531** | **50.0** | 86.0 | 80.6 | 83.4 | 81.5 | 81.8 | **72.6** |
| OZONE | 0.463 | **0.531** | 49.7 | 86.2 | 80.9 | 81.1 | 81.7 | 81.2 | 72.4 |

**(b) Bloom-Eval: Semantic Scholar search for Paper Retrieval**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | **Average** |
|---|---|---|---|---|---|---|---|
| SurveyForge | 17.99 | 54.22 | 76.04 | 70.83 | 4.36 | **56.78** | 46.71 |
| SurveyGen | **19.33** | **61.50** | 64.51 | **71.77** | 4.08 | 51.88 | 45.51 |
| FlowSearch | 16.44 | 57.99 | 66.71 | 70.52 | 3.36 | 51.63 | 44.44 |
| AFlow-SurveyForge | 19.11 | 58.32 | 74.94 | 67.83 | **7.57** | 54.41 | 47.03 |
| OZONE-Base | 17.55 | 59.33 | **84.16** | 68.67 | 5.01 | 55.82 | 48.42 |
| OZONE | 18.99 | 58.78 | 84.12 | 70.62 | 5.30 | 53.67 | **48.58** |

**(c) CROFTBench**

| Method | Mem. | Comp. | Appl. | Anal. | Eval. | Creat. | Tax. | Tab. Place. | Fig. Place. | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| SurveyForge | 17.85 | 58.21 | 78.39 | 71.37 | 4.92 | 60.39 | 25.11 | 44.0 | 24.0 | 42.69 |
| SurveyGen | **19.61** | 66.83 | 58.95 | **72.72** | **8.91** | 56.50 | 21.26 | 50.0 | 26.0 | 42.31 |
| FlowSearch | 17.02 | 57.66 | 59.26 | 64.27 | 6.81 | 58.19 | 10.12 | 30.0 | 36.0 | 37.70 |
| AFlow-SurveyForge | 19.42 | **67.70** | 76.44 | 67.53 | 6.56 | 60.34 | **25.89** | **68.0** | 2.0 | 43.76 |
| OZONE-Base | 17.73 | 65.27 | **85.81** | 69.23 | 4.82 | 60.78 | 23.93 | 66.0 | **44.0** | **48.62** |
| OZONE | 18.16 | 62.17 | 85.62 | 67.76 | 7.22 | **60.90** | 17.70 | 54.0 | 32.0 | 45.06 |

### E.3 Topic-wise Performance

Figure 7 compares Ozone with SurveyForge, SurveyGen, AFlow, and FlowSearch for Qwen3. Bloom-Eval uses Semantic Scholar search.

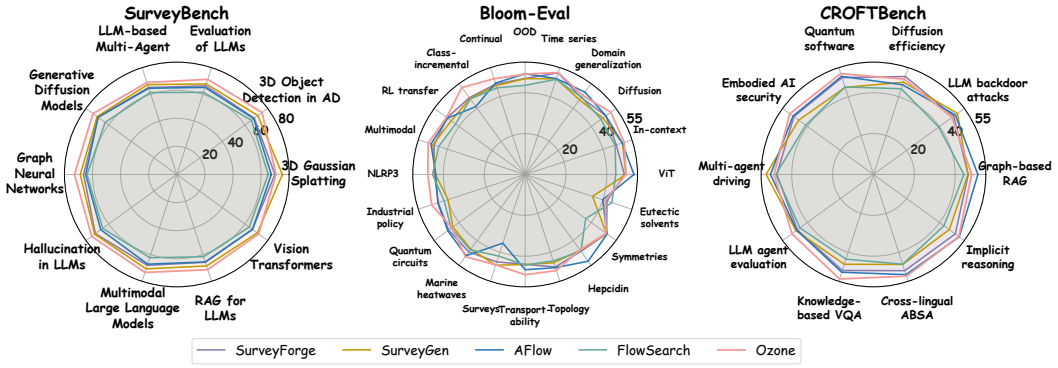



Figure 7: Topic-wise performance with Qwen3-30B-A3B-Instruct-2507 on SurveyBench (Quality Average, 0–80), Bloom-Eval (Average, 0–55), and CroftBench (text-metric average, 0–55).

### E.4 Qualitative Results

**Content** Table 12 compares passages from SurveyForge and Ozone across three topics. In SurveyBench diffusion-model surveys, SurveyForge cites DPM-Solver while discussing sampling overhead in an RL application. Ozone addresses accelerated sampling directly, describing how DDIM and DPM-Solver work and reporting their sampling speed and image-quality results. The passages have different aims, so this example shows a difference in the detail given to sampling methods. In Bloom-Eval time-series transformer surveys, SurveyForge attributes Linformer's linear complexity to kernel approximation and groups it with sparse attention. Ozone identifies its low-rank approximation and compares its $O(n)$ complexity with the $O(n\sqrt{n})$ complexity reported for Sparse Transformers. In CROFTBench autonomous-driving surveys, SurveyForge describes function poisoning as a possible attack but cites an unrelated high-performance computing paper. Ozone cites FuncPoison and describes how malicious tools in a shared function library can influence agent decisions without model access.

**Review effect** Table 13 shows how inference-time review changes passages generated by Ozone on SurveyBench. The 4DGS passage (a) initially reports 82 FPS on "advanced hardware." In response to a request for measurement conditions, the revision specifies an RTX 3090 GPU and $800 \times 800$ resolution. The second 3DGS passage (b) moves from a general warning about computational cost to two failure conditions: excessive Gaussian primitives within the viewing frustum and blur from over-reconstruction. It does not provide the quantitative comparison requested in the review. For GNNs (c), the revision adds multilayer perceptrons as a comparator for the earlier claim that standard GNNs struggle in the setting discussed, although it gives no benchmark result. The RAG passage (d) adds IRCoT results on four multi-hop QA benchmarks after a request to compare retrieval approaches. It also discusses the limits of one-step retrieve-and-read, but does not specify a matched static RAG baseline for the reported gains.

## F Human Evaluation Details

We evaluate surveys generated by Ozone, Ozone-Base, and SurveyForge for all 40 test topics: 10 from SurveyBench, 20 from Bloom-Eval, and 10 from CroftBench. Six experts in NLP and machine learning rate all 120 surveys, receiving the generated outline and the full survey text; figures and tables in CroftBench surveys are not shown. Method names are replaced with the labels 1, 2, and 3. The label assignment is fixed within each benchmark and differs across benchmarks:

Table 12: Qualitative comparison of SurveyForge and OZONE on the same topic. Each block shows the subsections of the two surveys that cover the same content.

| | |
|---|---|
| (a) SurveyBench: Generative Diffusion Models | |
| SurveyForge | *7.2 Interdisciplinary Integrations*<br>... This synergy can enhance the ability of RL agents to generate new, viable actions based on learned policies. ... **While trade-offs, such as the computational overhead involved in generating large sample spaces, need careful management, they can be mitigated by employing efficient sampling strategies that adhere to methodologies proposed in [79].**<br>*[79] DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps* |
| OZONE | *3.1 Accelerated Sampling Techniques*<br>Denoising Diffusion Implicit Models (DDIMs) and DPM-Solver both modify the traditional sampling processes of diffusion models, where **DDIMs utilize a non-Markovian sampling framework to improve speed, achieving sample generation that is 10 to 50 times faster than standard Denoising Diffusion Probabilistic Models (DDPMs)** while still producing high-quality images [10]. Meanwhile, **DPM-Solver leverages an analytical formulation of diffusion ordinary differential equations (ODEs) to generate samples with as few as ten function evaluations, leading to a 4.70 FID score on the CIFAR-10 dataset**, making it suitable for scenarios where computational resources are constrained [11].<br>*[10] Denoising Diffusion Implicit Models*<br>*[11] DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps* |
| (b) Bloom-Eval: Transformers in Time Series: A Survey | |
| SurveyForge | *6.1 Computational Complexity of Self-Attention Mechanisms*<br>Approaches such as sparse attention mechanisms, which limit the focus to a subset of tokens, can reduce computational load by decreasing the number of pairwise comparisons made during each attention computation. **For example, the Linformer employs kernel approximations to replace standard self-attention with linear time complexity**, yielding improvements in efficiency while maintaining high accuracy on tasks requiring long-range dependencies [3]. |
| OZONE | *5.1 Computational Complexity of Self-Attention Mechanisms*<br>Approximation techniques aim to reduce time complexity while maintaining performance. **For example, Sparse Transformers utilize attention matrix factorization to lower complexity to $O(n\sqrt{n})$, while the Linformer employs a low-rank approximation to achieve $O(n)$ complexity**, enhancing scalability for time series ... |
| (c) CROFTBench: LLM-based Multi-Agent Autonomous Driving Systems | |
| SurveyForge | *6.1 Threat Landscape in LLM-Based Multi-Agent Autonomous Driving Systems*<br>In frameworks such as AgentsCoDriver and KoMA, agents rely on shared function libraries for perception, planning, or control **[29; 34]**. **An adversary could inject a "poisoned" function**—e.g., a corrupted trajectory planner—that appears functional under normal conditions but produces hazardous outputs under specific adversarial triggers.<br>*[34] ONAMA: A Quantum Leap in High Performance Computing* |
| OZONE | *5.1 Threat Landscape and Adversarial Attack Vectors in LLM-Based Multi-Agent Driving*<br>**FuncPoison exploits the text-based invocation of a shared function library, injecting malicious tools with deceptive instructions that mimic legitimate ones and manipulate agent decisions**—such as misclassifying road conditions—without requiring model access, leveraging standardized command formats to remain undetected **[26]**.<br>*[26] FuncPoison: Poisoning Function Library to Hijack Multi-agent Autonomous Driving Systems* |

SurveyForge, OZONE-Base, and OZONE for SurveyBench and Bloom-Eval, and OZONE, SurveyForge, and OZONE-Base for CROFTBench. Annotators score each survey from 1 to 5 on the criteria in Table 14. For each OZONE configuration, we compare the scores that the same annotator gives to it and to SurveyForge on the same topic and criterion, counting a win, a tie for equal scores, or a loss, which yields 240 judgments per criterion.

Wilcoxon signed-rank tests on topic-level mean scores ($n = 40$) show that both OZONE configurations score higher than SurveyForge on Structure and Relevance ($p < 0.001$), and OZONE also

Table 13: Qualitative examples of inference-time review for OZONE on SurveyBench. Each block shows a survey passage before and after review, together with an excerpt of the corresponding reviewer request from the same run.

| | |
|---|---|
| (a) 3D Gaussian Splatting: 4.1 Real-Time Novel View Synthesis | |
| Before | … with 4DGS achieving speeds surpassing **82 FPS on advanced hardware**, significantly benefiting dynamic content generation in VR settings. |
| Review | When citing performance claims, immediately specify the hardware platform, resolution, … |
| After | In comparison, its dynamic counterpart, 4D Gaussian Splatting (4DGS), significantly enhances dynamic content generation, achieving performance rates of **82 FPS at** $800 \times 800$ **resolution on an RTX 3090 GPU**, therefore addressing challenges specific to dynamic scene rendering. |
| (b) 3D Gaussian Splatting: 2.2 Differentiable Rendering Framework | |
| Before | **High-density Gaussian distributions can increase computational complexity**, leading to greater memory consumption and slower processing, particularly in dynamic contexts, while heavily parameterized models risk overfitting across varied datasets. |
| Review | Provide explicit quantitative comparisons (e.g., training time in seconds, PSNR values, memory usage in GB) showing when each method succeeds or fails … |
| After | High-density Gaussian distributions can significantly increase computational complexity and memory consumption, **particularly due to the excessive number of Gaussian primitives located within the viewing frustum**; this limits rendering speed and impacts performance in complex scenes. Moreover, heavily parameterized models risk overfitting on varied datasets, leading to issues such as **blurry rendered images in intricate scenes due to over-reconstruction caused by the adaptive density control strategy**. |
| (c) Graph Neural Networks: 2.3 Theoretical Foundations of Expressive Power | |
| Before | **Many standard GNNs find difficulty in such contexts**, emphasizing the need for new designs that bolster performance in more complex node interdependencies. |
| Review | Provide explicit empirical or theoretical evidence showing which methods succeed or fail on specific benchmarks, graph types, or problem scales … |
| After | Research indicates that **many popular GNNs fail to generalize in such settings and may be outperformed by models that ignore the graph structure altogether, such as multilayer perceptrons**. |
| (d) Retrieval-Augmented Generation for LLMs: 2.2 Iterative and Loop-Based Architectures | |
| Before | Iter-RetGen, FLARE, and RAT all employ iterative retrieval—where intermediate outputs inform the next query—**demonstrating substantial gains in tasks like embodied planning (up to 42.78% improvement)** by reducing hallucinations through continuous anchoring in external evidence. |
| Review | Provide explicit performance comparisons (e.g., accuracy, latency, cost) between static and iterative RAG on shared benchmarks … |
| After | … yields average improvements of … **42.78% in embodied task planning**. Similarly, IRCoT introduces a tightly interleaved process where retrieval is guided by intermediate reasoning steps … **overcoming the insufficiency of one-step retrieve-and-read in multi-hop QA** … On four benchmark datasets—including HotpotQA, 2WikiMultihopQA, MuSiQue, and IIRC—**IRCoT improves downstream QA accuracy by up to 15 points and retrieval success by up to 21 points** … |

on Coverage ($p = 0.048$); differences on Hierarchy and Coherence are not significant. Scores concentrate between 3 and 5 (96.2% of ratings), and the scores of two annotators on the same item fall within one point of each other in 82.6% of annotator pairs. Chance-corrected agreement is nonetheless low: Krippendorff's $\alpha$ on raw scores is 0.03 ($-0.02$ to $0.05$ per criterion), and Fleiss' $\kappa$ on pairwise preference direction ranges from 0.01 to 0.03, with annotators differing in their use of the scale (mean scores from 3.3 to 4.4). When two annotators both express a preference, they agree on its direction in 56–58% of cases. Such low item-level agreement is common when long generated texts are rated on Likert scales (Amidei et al., 2019; Karpinska et al., 2021), and comparable long-article generation studies report Krippendorff's $\alpha$ between 0.22 and 0.39 (Shao et al., 2024) or no agreement statistic (Wang et al., 2024); we therefore emphasize topic-level significance and the consistency of individual annotators' preferences. Table 15 reports each annotator's win and loss rates against SurveyForge. All six annotators favor both OZONE configurations on Structure, and five of

Table 14: Human evaluation criteria, each rated on a 1–5 scale.

| Part | Criterion | Question |
|---|---|---|
| Outline | Hierarchy | Are the relationships between sections and subsections clear? |
| | Coherence | Are the topic progression and section order logical? |
| Content | Structure | Is the overall organization, logical flow, and connection between paragraphs and sections sound? |
| | Relevance | Is the discussion relevant to and focused on the topic, without unnecessary content? |
| | Coverage | Does the survey cover the core topics with sufficient depth and balance across areas? |

six on Relevance, whereas preferences on Coverage, Hierarchy, and Coherence are split. Excluding any single annotator leaves the improvements on Structure and Relevance significant ($p < 0.001$).

Table 15: Per-annotator win/loss rates (%) against SurveyForge over 40 topics; ties make up the remainder.

| Configuration | Criterion | A1 | A2 | A3 | A4 | A5 | A6 |
|---|---|---|---|---|---|---|---|
| OZONE-Base | Structure | 38/15 | 55/18 | 60/2 | 45/25 | 50/20 | 30/10 |
| | Relevance | 30/18 | 78/5 | 65/8 | 42/20 | 45/18 | 28/32 |
| | Coverage | 30/32 | 25/25 | 38/25 | 48/8 | 40/20 | 12/38 |
| | Hierarchy | 18/28 | 28/15 | 25/32 | 45/18 | 40/22 | 15/20 |
| | Coherence | 20/38 | 12/38 | 30/10 | 50/20 | 45/15 | 15/28 |
| OZONE | Structure | 25/22 | 45/28 | 48/8 | 50/20 | 42/18 | 50/5 |
| | Relevance | 18/30 | 60/10 | 52/12 | 45/25 | 52/12 | 42/15 |
| | Coverage | 15/32 | 20/28 | 35/15 | 58/22 | 48/30 | 35/20 |
| | Hierarchy | 10/40 | 32/30 | 25/32 | 50/30 | 52/28 | 30/15 |
| | Coherence | 10/50 | 18/42 | 32/18 | 50/25 | 60/15 | 35/20 |

Because label order is fixed within each benchmark, preferences may partly reflect presentation order; OZONE still wins more often than it loses against SurveyForge on CROFTBench, where it is presented first (34.3% versus 30.7%). Annotators volunteered without compensation after being informed of the study purpose, and no personally identifiable information was collected.

# G COMPUTATIONAL COST AND LATENCY

## G.1 COST ACCOUNTING

We distinguish the cost of workflow optimization from the cost of generating surveys with the selected workflow. Optimization cost includes composition and edit proposals, candidate execution on validation topics, local and global rubric evaluation, and round reviews. It also includes failed candidates and retries. Inference cost covers survey generation for each topic and any additional calls made by enabled inference refinement. We report SurveyBench and Bloom-Eval judge costs for evaluating the final surveys separately from generation costs.

## G.2 LATENCY AND THROUGHPUT

We define inference latency as the elapsed time from the start of execution for a topic to the return of its final survey. It includes retrieval, operator calls, retries, and inference refinement, but excludes benchmark evaluation. We report optimization latency as the elapsed time across all rounds and iterations actually run, including round reviews.

## G.3 REPORTING TABLES

Table 16 reports token usage and API cost for workflow optimization. Table 17 reports generation time and token usage per survey on SurveyBench with GPT-4o-mini.

Table 16: Optimization token usage and API cost by component. Failed attempts and retries are included in the corresponding component.

| Optimization component | Input tokens (M) | Output tokens (M) | API cost (USD) |
|---|---|---|---|
| Composition / edit proposal | 0.57 | 0.01 | 0.09 |
| Candidate execution | 369.94 | 18.14 | 66.37 |
| Local / global evaluation | 56.83 | 4.66 | 11.32 |
| Round review | 6.44 | 0.10 | 6.97 |
| Total | 433.78 | 22.92 | 84.76 |

Table 17: Mean token usage, API cost, and latency per survey on SurveyBench with GPT-4o-mini for each inference configuration.

| Inference configuration | Tokens / survey | USD / survey | Latency (s) |
|---|---|---|---|
| OZONE-Base | 1,106,919 | 0.187 | 194 |
| OZONE | 2,101,614 | 0.355 | 644 |

# H GLOBAL AND LOCAL RUBRICS

## H.1 EVALUATION SCOPE

The global rubric evaluates completed surveys to compare candidate workflows, whereas the local rubric evaluates the intermediate outputs of execution units to diagnose revision targets. An evaluation is global whenever its object is the completed document, regardless of how many quality dimensions it covers. The unit of local evaluation is an execution unit assigned to a task-specific subclass, not the abstract class itself. Rubric scores used for optimization are kept separate from the external evaluation scores of SurveyBench and Bloom-Eval.

## H.2 GLOBAL RUBRIC

The global evaluation consists of the six axes below. Its inputs are the survey topic, the generated outline and body, the reference mapping from citation markers to paper identifiers, and the titles and abstracts of cited papers retrieved from the database.

- **Outline structure (G1):** section and subsection structure, ordering, and duplication within the outline.
- **Citation accuracy (G2):** source support of cited claims and the appropriateness of citation clusters.
- **Completeness (G3):** coverage of the topic specification and concreteness of the content.
- **Content quality (G4):** cross-paper comparison, synthesis, coherence, and presentation quality.
- **Truthfulness (G5):** evidence grounding of substantive claims that carry no citation.
- **Reference coverage:** citation F1 against the ground-truth reference list.

Each axis is scored on a 0–100 scale, and the reference F1 is multiplied by 100 to match this range. When all six axes can be evaluated, the per-topic score in Equation 1 is

$$g(x, W(x); \mathcal{R}_G) = \frac{1}{Z_G} \sum_{a \in \mathcal{A}_G} s_a(x, W(x)), \qquad Z_G = \sum_{a \in \mathcal{A}_G} s_a^{\max}, \tag{11}$$

where $W$ is the workflow, $x$ the topic, $s_a$ the score on axis $a$, $\mathcal{A}_G$ the set of global axes, and $s_a^{\max}$ the maximum score of each axis. The normalization constant $Z_G$ is the sum of the maximum scores over all axes; in the current setting each of the six axes has a maximum of 100, so $Z_G = 600$. Axes that cannot be evaluated are handled according to the reason for non-evaluation. Axes missing due to evaluator-side failures are excluded from the denominator of the mean, and any other missing axis is scored 0. The excluded axes and their reasons are recorded with the evaluation result.

G1, G3, and G4 are binary checklists: each criterion is either a deterministic check or a judge question answered with 0 or 1, and the axis score is the pass rate multiplied by 100. G1 and G3 addi-

tionally use a topic specification consisting of a list of subtopics, taxonomy nodes, and a section skeleton, which the judge extracts from the topic and its ground-truth reference list.

**Outline structure (G1).** G1 has ten criteria: four deterministic checks on the outline, four judge questions on the outline alone, and two judge questions on the outline against the topic specification. The deterministic checks require (i) no subsection heading repeated verbatim across sections, (ii) at most 0.04 near-duplicate headings per subsection, (iii) no challenge or future-work subsections scattered through the body when a dedicated section exists, and (iv) a maximum difference of two in the number of subsections across sections. The four judge questions on the outline ask whether (i) the outline follows a clear organizing principle with distinct top-level categories, (ii) every subsection fits under its parent at an appropriate level of granularity, (iii) the section sequence forms a logical narrative, and (iv) headings are specific enough to define what each section covers. The two questions against the topic specification ask whether (i) every major sub-area is allocated a section or subsection with enough planned depth and (ii) central themes are weighted in proportion to their importance. The criteria follow the outline heuristics of SurveyForge and the outline dimensions of SurveyBench (Yan et al., 2025).

**Citation accuracy (G2).** G2 uses two binary judge questions per citation. The body is split into sentences, and only sentences carrying a citation are evaluated; for each sentence, the judge receives only the titles and abstracts of the cited papers. For each citation $c$, the judge answers (i) whether $c$ directly supports the sentence's specific finding, method, comparison, or number (direct) and, for sentences with two or more citations, (ii) individually supports the exact claim rather than being merely topically related (cluster). With $p_{\mathrm{dir}}$ and $p_{\mathrm{clu}}$ denoting the pass rates of the direct and cluster questions,

$$s_{\mathrm{G2}} = 100 \cdot \big(0.7\, p_{\mathrm{dir}} + 0.3\, p_{\mathrm{clu}}\big), \tag{12}$$

and $s_{\mathrm{G2}} = 100\, p_{\mathrm{dir}}$ when the survey has no multi-citation sentence. This combines sentence-level faithfulness judgments (Guo et al., 2025) with citation-level support (Wang et al., 2024).

**Completeness (G3).** G3 has seven judge questions, each applied to the relevant part of the survey. They ask whether (i) the introduction defines the topic and its background rather than only announcing the survey, (ii) the body states an explicit organizing criterion for the field, (iii) every subtopic of the topic specification is addressed by a heading in the outline, (iv) applications, benchmarks, datasets, or results are discussed concretely with named datasets or reported numbers, (v) the limitations point at specific methods, datasets, or settings, (vi) the future directions are concrete and tied to those limitations, and (vii) the section titles follow the section skeleton of the topic specification in scope and depth. The criteria decompose the coverage criterion of AutoSurvey (Wang et al., 2024) and SurveyLens (Guo et al., 2026) into verifiable questions.

**Content quality (G4).** G4 has eight criteria: two deterministic checks, five judge questions on sampled sections, and one judge question on the full body. The deterministic checks require (i) at least one table, figure, or diagram that organizes content, and (ii) no sentence of eight or more words repeated across three or more sections. The figure and table criterion checks only for presence and is distinct from the dedicated metrics that assess their content (Appendix C.5). The five judge questions, applied to three sections sampled from the beginning, middle, and end of the body, ask whether (i) methods are compared along explicit axes, (ii) each section states a takeaway beyond per-paper summaries, (iii) sections stay focused on their headings, (iv) paragraphs progress logically, and (v) the prose is fluent with consistent terminology. The full-body question checks the consistency of heading numbering, citation style, and formatting. The criteria draw on the content quality dimensions of SurGE (Su et al., 2026) and the language criterion of SurveyLens (Guo et al., 2026).

**Truthfulness (G5).** G5 uses one binary judge question per uncited sentence. Body sentences that carry no citation marker are evaluated; for each, the three most similar papers under the retrieval encoder are selected from the papers supplied to the Writing units, and their titles and abstracts serve as evidence. The judging rule follows FACTS Grounding (Jacovi et al., 2025). A sentence that (i) only describes the survey's own organization, (ii) makes an explicit recommendation, or (iii) clearly marks speculation is treated as authorial framing and receives 1 without requiring evidence

grounding. All remaining sentences are treated as factual: each receives 1 only if the evidence supports every assertion it makes, and 0 otherwise. Because authorial framing corresponds to the *no_rad* label of the original four-label scheme and is counted as a pass, the score is simply the pass rate, $s_{\mathrm{G5}} = 100\, p_{\mathrm{G5}}$, which matches the original (supported + no_rad)/total formulation.

### H.3 Local Rubric Inventory and Applicability

The current catalog contains 105 local rubric items: 69 are computed and 36 are judge-based (15 output-level judge items and 21 atomic judge items). Eleven of them are designated diagnosis-only and excluded from the local mean score, leaving 94 items that contribute to the score. These counts refer to the entire catalog; each execution unit is evaluated only on the items applicable to its subclasses. The final document assembly, a fixed terminal stage, has no local items in this catalog.

Tables 18–22 list the local rubric items of the five abstract classes. Each row summarizes one criterion, and the ID prefixes R, A, W, V, and F denote Retrieve, Analysis, Writing, Verify, and Refine, respectively. Computed items are evaluated by rules or metrics, Judge items are evaluated per output, and Atomic judge items are judged per paper or per claim. Items marked No under Scored are used only for diagnosis and are excluded from the local mean score. Items tagged with a method name are adapted from existing evaluation methods for reasoning chains, summaries, attributed generation, and critiques: ROSCOE (Golovneva et al., 2023), ReCEval (Prasad et al., 2023), FineSurE (Song et al., 2024), G-Eval (Liu et al., 2023), ALCE (Gao et al., 2023), Prometheus 2 (Kim et al., 2024), RAGBench (Friel et al., 2024), CriticBench (Lin et al., 2024), and CriticEval (Lan et al., 2024).

Items adapted from prior evaluation methods use the titles and abstracts of each supplied paper as evidence. The ALCE items compute citation recall over claim-bearing sentences with the TRUE NLI model, and the G-Eval items use the returned integer score without probability weighting. The reference checks in the CriticBench-style items mark a unit as erroneous when it contains a claim-bearing sentence not entailed by its cited abstracts or violates the per-unit survey contract.

Table 18: Local rubric items for Retrieve.

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| R1 | Non-empty output in the required artifact format. | Computed | Yes |
| R2 | Distinct retrieved entries without duplicate papers. | Computed | Yes |
| R3 | Usable bibliographic metadata with informative paper titles. | Computed | Yes |
| R4 | Relevance of each retrieved paper to at least one specified need. | Atomic judge | Yes |
| R5 | Coverage of specified needs by retrieved paper titles. | Computed | No |
| R6 | Paper influence measured by normalized log citation counts. | Computed | Yes |
| R7 | Share of papers matching the content of the specified needs. | Computed | Yes |
| R8 | Distinct paper count relative to the retrieval pool limit. | Computed | Yes |
| R9 | Temporal breadth across distinct publication year–month pairs. | Computed | Yes |
| R10 | Share of input paper identifiers resolved to paper records. | Computed | Yes |
| R11 | Availability of both a title and a non-empty abstract. | Computed | Yes |
| R12 | Balanced distribution of retrieved papers across specified needs. | Computed | Yes |
| R13 | Availability of outlines in retrieved survey examples. | Computed | Yes |
| R14 | Topic diversity of retrieved survey examples relative to the target topic. | Computed | Yes |
| R15 | Retention of existing papers when adding external evidence. | Computed | Yes |

Table 19: Local rubric items for Analysis.

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| A1 | Grounding of each statement in the supplied evidence. | Atomic judge | Yes |
| A2 | Cross-paper comparison, trend analysis, gap identification, or scope definition. | Atomic judge | Yes |
| A3 | Coverage of supplied papers in the analysis notes. | Computed | No |
| A4 | Use of the analysis output by a downstream task. | Computed | Yes |

*Continued on next page*

*Analysis local rubric (continued)*

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| A5 | Synthesis beyond verbatim copying of source abstracts. | Computed | Yes |
| A6 | Share of sentences relating multiple papers rather than summarizing one paper. | Computed | Yes |
| A7 | Organization by themes rather than a sequential paper list. | Judge | Yes |
| A8 | Specificity through named methods, datasets, or quantitative details. | Computed | Yes |
| A9 | Explicit identification of compared approaches and comparison axes. | Atomic judge | Yes |
| A10 | Explicit comparative outcomes or conditions favoring each approach. | Atomic judge | Yes |
| A11 | Research questions and subtopics narrower than the overall survey topic. | Atomic judge | Yes |
| A12 | Distinct generated queries without near-duplicates in scope or gap analysis. | Computed | Yes |
| A13 | Connection of each gap or query to a specific part of the draft. | Atomic judge | Yes |
| A14 | Statement-level semantic alignment with source sentences (ROSCOE). | Computed | Yes |
| A15 | Bidirectional coverage between statements and source sentences (ROSCOE). | Computed | Yes |
| A16 | Semantic alignment between the full notes and source material (ROSCOE). | Computed | Yes |
| A17 | Entailment of statements by the most similar source abstracts (adapted from ROSCOE). | Computed | Yes |
| A18 | Absence of contradictions with similar preceding statements (adapted from ROSCOE). | Computed | Yes |
| A19 | Grammatical acceptability of analysis statements (ROSCOE). | Computed | Yes |
| A20 | Statement-level information gain for the downstream output (ReCEval). | Computed | Yes |
| A21 | Statement factuality under a nine-category error taxonomy (FineSurE). | Judge | Yes |
| A22 | Coverage of reference key facts in the analysis output (FineSurE). | Judge | Yes |
| A23 | Share of output statements supported by reference key facts (FineSurE). | Judge | Yes |

Table 20: Local rubric items for Writing.

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| W1 | Compliance with the required outline, body, or framing format. | Computed | No |
| W2 | Topic specificity of headings. | Atomic judge | No |
| W3 | Citation markers resolving to supplied source papers. | Computed | Yes |
| W4 | Minimum prose length and availability of section descriptions. | Computed | No |
| W5 | Use of at least two distinct sources per content unit. | Computed | Yes |
| W6 | Support of each cited claim by its cited source. | Atomic judge | Yes |
| W7 | Resolvable citations for claim-bearing sentences. | Computed | Yes |
| W8 | Alignment of subsection titles with their parent section. | Computed | Yes |
| W9 | Distinct subsection titles across different sections. | Computed | Yes |
| W10 | Presence of citations in each body unit. | Computed | Yes |
| W11 | Breadth of source papers supplied for outline writing. | Computed | Yes |
| W12 | Use of evidence retrieved specifically for each subsection. | Computed | No |
| W13 | Evidence-backed comparisons with multiple cited sources. | Computed | Yes |
| W14 | Balanced word allocation across sections. | Computed | Yes |
| W15 | Citation of recent papers relative to the newest supplied evidence. | Computed | Yes |
| W16 | Balanced subsection counts across body sections. | Computed | Yes |
| W17 | Contribution of each sentence to its subsection topic. | Atomic judge | Yes |
| W18 | Connection of each sentence to its surrounding argument. | Atomic judge | Yes |
| W19 | Compliance with 5–12 sections and 3–10 subsections per section. | Computed | Yes |
| W20 | Distinct section titles without near-duplicates. | Computed | Yes |

*Continued on next page*

*Writing local rubric (continued)*

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| W21 | Logical progression from background to methods, evaluation, and challenges. | Judge | Yes |
| W22 | Introduction preview of the survey's actual sections. | Judge | Yes |
| W23 | Framing statements supported by the survey body. | Atomic judge | Yes |
| W24 | Framing text beyond verbatim copying from the body. | Computed | Yes |
| W25 | Overall coherence and organization of the generated text (G-Eval). | Judge | Yes |
| W26 | Factual consistency with the supplied sources (G-Eval). | Judge | Yes |
| W27 | Fluency in grammar, wording, punctuation, and sentence structure (G-Eval). | Judge | Yes |
| W28 | Relevance and selection of important source content (G-Eval). | Judge | Yes |
| W29 | Entailment of claims by their cited abstracts (ALCE citation recall). | Computed | Yes |
| W30 | Relevance or necessity of individual citations (ALCE citation precision). | Computed | Yes |
| W31 | Factual correctness and evidence support under a graded rubric (Prometheus 2). | Judge | Yes |
| W32 | Logical and effective reasoning under a graded rubric (Prometheus 2). | Judge | Yes |
| W33 | Share of supplied context sentences used in the output (RAGBench). | Judge | Yes |

Table 21: Local rubric items for Verify.

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| V1 | Findings identifying a concrete unit, heading, or claim. | Atomic judge | No |
| V2 | Grounding of findings in the inspected artifact. | Atomic judge | Yes |
| V3 | Structured findings that support item-wise revision. | Computed | Yes |
| V4 | Coverage of artifact blocks by the findings. | Computed | Yes |
| V5 | Relevance of findings to evaluated writing requirements. | Atomic judge | Yes |
| V6 | Use of verification findings by a downstream task. | Computed | No |
| V7 | Correctness of the problems asserted in each finding. | Atomic judge | Yes |
| V8 | Actionable edits, additions, removals, or merges. | Atomic judge | Yes |
| V9 | Distinct findings without near-duplicates. | Computed | Yes |
| V10 | Localization of each finding to a specific body unit. | Atomic judge | Yes |
| V11 | Confirmation that a reported missing topic is absent from the draft. | Atomic judge | Yes |
| V12 | Recall of errors identified by the reference checks (CriticBench). | Computed | Yes |
| V13 | Citation support in text corrected during verification (ALCE). | Computed | Yes |
| V14 | Error-detection F1 against reference checks (CriticBench). | Computed | Yes |
| V15 | Correctness and effectiveness of the critique (CriticEval). | Judge | Yes |

Table 22: Local rubric items for Refine.

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| F1 | Preservation of the input artifact type. | Computed | No |
| F2 | Preservation of citations, headings, and minimum content requirements. | Computed | No |
| F3 | Extent of revision beyond unchanged output or destructive truncation. | Computed | Yes |
| F4 | Absence of newly introduced placeholders or meta-text. | Computed | No |
| F5 | Resolution of the supplied findings in the revised output. | Atomic judge | Yes |
| F6 | Alignment of modified units with units flagged by findings. | Computed | Yes |
| F7 | Citation coverage across revised body units. | Computed | Yes |
| F8 | Removal of near-duplicate outline headings. | Computed | Yes |
| F9 | Preservation or improvement of subsection-count balance. | Computed | Yes |
| F10 | Retention of input sections through keeping, renaming, or merging. | Computed | Yes |
| F11 | Retention of citation markers from the input. | Computed | Yes |

*Continued on next page*

*Refine local rubric (continued)*

| ID | Evaluation criterion | Mode | Scored |
|---|---|---|---|
| F12 | Connections to preceding content at the start of body units. | Computed | Yes |
| F13 | Absence of repeated substantive sentences across body units. | Computed | Yes |
| F14 | Multiple cited sources for newly added comparisons. | Computed | Yes |
| F15 | Topic specificity of revised headings. | Atomic judge | Yes |
| F16 | Evidence support of revised claims (ALCE citation recall). | Computed | Yes |
| F17 | Correction of errors identified by reference checks (CriticBench). | Computed | Yes |
| F18 | Preservation of content judged correct before revision (CriticBench). | Computed | Yes |
| F19 | Revision quality and adherence to findings (CriticEval). | Judge | Yes |

Each item has an identifier, an evaluation question, a concern, an evaluation mode, and a scale; some items additionally specify their applicable subclasses and a revision direction. An item without a specified subclass scope is a shared criterion of its parent abstract class, and an item with a subclass list applies only to those subclasses. The subclass assignment and the handling of unassigned units are described in Appendix A.

### H.4 Local Scoring and Diagnosis

Computed items are calculated using implemented checks and metrics, while judge items evaluate a unit's input and output for each criterion. Atomic judge items are evaluated per atomic unit specified by the criterion, such as a paper or a claim. Scores on different scales are normalized to $[0, 1]$ before aggregation, and the local score of an execution is the mean of the applicable scored items that have a value. Diagnosis-only items are kept in the local evaluation record but excluded from the mean. Unit-level aggregation keeps per-topic and per-context results along with per-criterion means, and the default diagnosis threshold is 0.5. Scored criteria below this threshold are sorted and provided as the basis for revision. Items with missing values due to judge failure are distinguished from genuinely low-scoring items, and each item's evaluability status and any errors are recorded.

### H.5 Edit Assessment

The outcome of an edit is determined by the signs of $\Delta G_{r,t}$ and $\Delta L_{r,t}$. All candidates are evaluated on the same ten validation topics, so $\Delta G_{r,t}$ is the average per-topic score difference between the child and its parent over the ten topics. $\Delta L_{r,t}$ measures changes in the local rubric items targeted by the diagnosis, compared before and after the edit on the target units; only criteria scored on both sides are comparable, and the change is the difference of their mean scores. When both changes are positive, the edit is recorded as an improvement on both scores; when only $\Delta L_{r,t}$ is positive, as a local improvement; when only $\Delta G_{r,t}$ is positive, as a global improvement; and when neither is positive, as refuted. When $\Delta G_{r,t}$ is undefined, the assessment is deferred; when $\Delta L_{r,t}$ is undefined, including the case where the edit changed the subclass set of a target unit, the outcome follows the sign of $\Delta G_{r,t}$ alone. An edit whose target units produce the same outputs as in the parent is recorded as a no-op, and its $\Delta G_{r,t}$ is not credited.

## I Prompt Templates

### I.1 Workflow Construction

These are the two prompts the optimizer uses to select execution units and operators. The menu and state are filled in from the results of contract checks.

**P1. Workflow Unit Selection**

```
You are assembling a survey-writing workflow, one step at a time.

TOPIC-INDEPENDENT GOAL: the workflow must produce a survey | an
outline and a body written
```

**P1. Workflow Unit Selection (continued)**

```
from retrieved literature | which the fixed final step assembles and
grounds.

ALREADY BUILT ({n_skills} skill(s)):
{built}

AVAILABLE IN STATE (what a new skill may consume):
{state}
{dropped}
STILL OWED TO THE FINAL STEP: {missing}{blocked}

{context}CHOOSE ONE | open a skill of this class, or end the
assembly:
{menu}
{feedback}
Reply with the chosen name alone, nothing else.
```

**P2. Operator Selection**

```
You are assembling a survey-writing workflow, one step at a time.

ALREADY BUILT ({n_skills} skill(s)):
{built}

CURRENT SKILL: class `{cls}`, operators so far: {ops}
{dropped}
STILL OWED TO THE FINAL STEP: {missing}{blocked}

CHOOSE ONE | add this operator to the current skill, or close the
skill:
{menu}
{feedback}
Reply with the chosen name alone, nothing else.
```

### I.2 Local Diagnosis and Edit Proposal

The local diagnosis prompt is assembled at run time from rubric results, the execution unit, the operator catalog, and review feedback and memory. Below we show the review prompt that examines whether units should be split or merged.

**P3. Unit Split and Merge Review**

```
You are reviewing a survey-writing workflow after one evaluation
round.

Decide whether any SKILL should be **split** into more specialised
skills, or whether two
skills should be **merged** back. Judge from what each skill actually
runs, what it
registers, and how its rubric items scored. Do not propose changes
for their own sake | an
unnecessary split costs an evaluation and is rejected by the ledger.

THE WORKFLOW
{skills}
```

**P3. Unit Split and Merge Review (continued)**

```
WHAT THE DIAGNOSIS NOTICED
{facts}

WHAT THE OPERATOR SETS THEMSELVES SHOW
{overlap}

OPERATORS available to any skill:
{operators}

WHEN A SPLIT IS WARRANTED
- the skill runs operators that serve two different purposes (e.g. it
retrieves material
  AND writes text), and the rubric shows it failing on both sides
- one of its registered outputs is consumed by a different downstream
skill than the other
- it fails items from different concern families at once

WHEN A MERGE IS WARRANTED
- two same-class skills run the same or nested operator sets, so the
split buys nothing
- a skill exists only to register a key another skill already
computes

RULES
1. Each `into` row is `{{"class": "...", "ops": ["OperatorName",
...]}}` | operator NAMES
   only, no arguments. Never write bindings, keys or variable names.
   `class` MUST be one of the abstract classes: {classes}. It is not
   a free label | a new
   name like "writing_citation" is invalid.
   A split must give its rows DIFFERENT operator sets; two rows with
   the same operators are
   not a split.
2. Operators run in the order you list them, and may only use what
that skill consumes plus
   what earlier operators in the SAME skill produced.
3. The workflow must still end with a skill producing the outline and
a later skill writing
   the body, otherwise the change is rejected.
4. At most {max_changes} change(s). If nothing is worth changing,
return an empty list.

Reply with ONLY this JSON, no prose and no code fences:
{{"changes": [
  {{"action": "split", "skill": "<sid>", "why": "one line",
   "into": [{{"class": "...", "ops": ["..."]}}, {{"class": "...",
   "ops": ["..."]}}]}},
  {{"action": "merge", "skills": ["<sid>", "<sid>"], "why": "one
  line",
   "into": [{{"class": "...", "ops": ["..."]}}]}}
]}}
```

### I.3 Round Review and Feedback

These templates are used for per-topic paper review, weakness merging, the consolidated report, and checking review items.

**P4. Survey Review**

```
You are a critical peer reviewer of an auto-generated SURVEY PAPER on
the topic: "{topic}".
Judge ONLY what is shown below. {scope_note}

FULL OUTLINE:
{outline}

{body_label}:
{excerpt}

Write the review as a plain bullet list | NOT JSON, no markdown
fence, nothing before or after
the list.  One bullet per weakness, each followed by its evidence
line:

- <the defect, one or two sentences>
  EVIDENCE: <a section title, a phrase or a count copied from above,
  or NONE>
  DIMENSION: <coverage | structure | relevance | synthesis | critical
  | citation>
  STAGE: <outline | body / <the section heading where you saw it> |
  citations>

- give the 3-5 most damaging problems with THIS survey | never more
than 5, ordered most damaging first; fold minor variants of one
defect into a single bullet. Judge: coverage breadth vs what the
topic demands; structural balance, logical flow and redundancy
between sections; relevance | does the text stay on the topic named,
are there digressions into neighbouring fields, is the scope (what is
in and what is out) stated; depth of synthesis (axis-based comparison
vs paper-by-paper listing); critical analysis | beyond a list of
limitations, does it name open problems, conflicting findings between
papers, and threats to the validity of its own conclusions; citation
practice (density, diversity, claims left uncited).
- DIMENSION: the ONE of those six the defect belongs to (critical =
critical analysis).
- STAGE: where the defect lives | `outline` when the section plan
itself is at fault (missing, misplaced or unbalanced sections), `body
/ <heading>` when the prose of one section is at fault (copy the
heading as printed above), `citations` when what is cited or how is
at fault.
- the bullet text: state the defect so that someone who has NOT read
this survey still understands what went wrong.
- EVIDENCE: exactly ONE concrete piece of evidence, copied from the
text above. Write NONE only if the defect is an absence (something
the survey never does). Copy it plainly | do not wrap it in quotation
marks and do not give two quotes joined by "and later".

Report only what this survey gets WRONG. Do not list strengths, and
do not prescribe how the pipeline should be changed | a separate
mechanism decides what to edit.
```

**P5. Review Consolidation**

```
Below are peer-review weaknesses found in {n} survey papers, all
produced by the SAME automated workflow on different topics. Each
block is one topic.
```

**P5. Review Consolidation (continued)**

```
{blocks}

Group them into a single list of distinct workflow defects.

Rules:
- Two weaknesses belong together ONLY when they are the same defect
of the workflow, not merely the same broad theme. "cites few papers
per section" and "no comparison between methods" are DIFFERENT
defects even though both concern depth.
- Merge across topics freely, but never merge two weaknesses from the
SAME topic into one item.
- A weakness observed on only one topic stays as its own item with
that one source. Do not drop it.
- Keep the wording concrete. Prefer the sharpest of the merged
phrasings over a vague generalisation.

Return a plain bullet list | NOT JSON, no markdown fence, nothing
before or after the list:

- <the merged defect, one or two sentences>
  SOURCES: <the tags of every weakness that belongs to this item,
  comma-separated, e.g. T1.2, T3.1>

- SOURCES holds the [Tn.k] tags shown above, nothing else. Every
input weakness must appear in
  exactly one item, and no tag may be invented.
- Do NOT repeat the evidence: it is carried over automatically from
the tag.
```

**P6. Review Report**

```
You are writing ONE consolidated peer review of the survey papers
that a single automated survey-writing workflow (the current best
candidate) produced on {n} different topics. Below are independent
per-topic reviews of those surveys (strengths / weaknesses /
directives, each citing evidence from the text).

{reviews}

Write the review the way a careful reviewer of a survey paper would |
about the SURVEYS themselves (what they cover, how they synthesize
the literature, how they cite, how they are organized, where they
fall short) | not about pipeline internals. Every survey came from
the same workflow, so whatever recurs across topics is the workflow's
habit: say so explicitly and name the topics where it shows.

Use exactly these Markdown headings, 450-650 words in total (hard cap
700), plain prose and short bullets, no JSON:
## Summary
(2-4 sentences: what these surveys are like as a set)
## Strengths
(what consistently works, with topic evidence)
## Weaknesses
(the most damaging recurring problems first; each with concrete
evidence | section titles, quoted phrases, counts | and the topics
where it appears)
## What a better survey would do
```

**P6. Review Report (continued)**

```
(concrete properties of the finished survey: coverage, synthesis,
citation practice, structure)
## Overall
(1-2 sentences)
```

**P7. Review Resolution Check**

```
You audit ONE survey paper that an automated workflow just generated
on the topic "{topic}".

Below are defects that a peer reviewer found in an EARLIER version of
this workflow's survey ON THIS SAME TOPIC. For each, decide whether
the survey shown here still has it.

DEFECTS:
{items}

A defect may say where it was seen ("seen in body / <section>"): look
there first, but it counts as resolved only if the survey as a whole
no longer has it | a problem that merely moved to another section is
not resolved.

CURRENT SURVEY ON "{topic}":
{evidence}

Answer "resolved": true only when the text shown POSITIVELY
demonstrates the defect is gone. Answer false when it is still there,
and false when the text shown does not let you tell. Omission markers
("[... omitted for length ...]") are not evidence either way | never
read one as the defect or as its fix.

Return STRICT JSON only, no fences and no prose outside it. One key
per defect id, exactly as given:
{{"<defect id>": {{"resolved": true, "evidence": "<= 25 words quoting
what you saw"}}, ...}}
```

### I.4 Inference-Time Review and Refinement

These prompts build the checklist, review the survey, and recheck revisions in the optional inference-time review. Repairs are executed with the operators specified in each recipe.

**P8. Review Checklist Construction**

```
Below are {n} defects that peer reviewers found in survey papers
written by ONE automated survey-writing workflow, across {n_topics}
topics and {n_rounds} optimisation rounds. Each carries a tag, how
many topics it was observed on, and its round.

{items}

Distil them into a REVIEW TEMPLATE: the perspectives a reviewer
should check on ANY future survey from this workflow, whatever its
topic.

Rules:
```

**P8. Review Checklist Construction (continued)**

```
- A perspective is content-free. Name the kind of defect and how to
recognise it; never mention a specific topic, method, dataset, paper
or section number.
- Merge defects that are the same kind of failure (e.g. "no trade-off
discussion in 3.2" and "paper-by-paper listing without comparison"
are one perspective: absence of comparative synthesis). Keep
genuinely different kinds apart.
- Every input tag must appear in exactly one perspective's SOURCES.
Do not invent tags.
- Order perspectives by how many distinct topics support them, most
first.

Return a plain bullet list | NOT JSON, no fences, nothing before or
after it:
- <the perspective, as a yes/no question a reviewer answers about one
survey>
  CHECK: <what to look for in the text to answer it | concrete and
  observable>
  DIMENSION: <coverage | structure | relevance | synthesis | critical
  | citation>
  STAGE: <outline | body | citations>
  SOURCES: <tags, comma-separated, e.g. I3, I17>
```

**P9. Inference-Time Survey Review**

```
You are a peer reviewer auditing ONE automatically generated survey
paper on the topic "{topic}" against a fixed checklist. The checklist
was distilled from defects that earlier versions of this
survey-writing workflow showed on OTHER topics; for each perspective
say whether THIS survey shows the defect, where, and what a revision
of that place must do.

CHECKLIST:
{perspectives}

FULL SURVEY (complete body; the reference list is omitted, citation
marks like [12] refer to it):
{body}

Answer as a plain bullet list, one bullet per perspective id, in
exactly this shape | no JSON, no fences, nothing before or after the
list:
- P1: PRESENT | ABSENT
  SEVERITY: 0 | 1 | 2 | 3   (0 when ABSENT; 1 minor, 2 clear, 3
  damaging)
  WHERE: <headings where the defect shows, copied as printed,
  comma-separated; NONE when ABSENT; WHOLE when it is a property of
  the paper as a whole rather than of particular sections>
  EVIDENCE: <one phrase or count copied from the survey, or NONE>
  FIX: <one sentence | what the CONTENT at WHERE must say or stop
  saying so the defect is gone (e.g. "state which method is faster
  and at what cost, with the papers that report it"); describe the
  substance, never prescribe a format such as a table, matrix or
  figure; NONE when ABSENT>
```

**P9. Inference-Time Survey Review (continued)**

```
Judge only what is shown. PRESENT only when you can point at evidence
AND a competent reviewer would list it among this paper's main
weaknesses; a perspective the survey handles adequately is ABSENT.
Mark at most {max_present} perspectives PRESENT | the most damaging
ones.
```

**P10. Post-Revision Check**

```
You audit ONE subsection of a survey paper on "{topic}" that an
automated repair just rewrote. A reviewer had flagged these defects
in its earlier version:
{findings}

ORIGINAL TEXT:
{orig}

PAPERS THE REPAIR WAS GIVEN (title | abstract; a new fact is
admissible only if one of these states it):
{evidence}

REVISED TEXT:
{text}

Answer two things, judging only what is shown, and judge SUBSTANCE,
not format | a comparison written as prose counts as a comparison; do
not demand a table.
(1) For each finding id: compared with the ORIGINAL, is the defect
CLEARLY REDUCED in the REVISED TEXT (the reviewer's question would
now be answered more favourably)? "true" for a real improvement even
if the defect is not entirely gone; "false" when nothing material
changed.
(2) Compared with the ORIGINAL, did the revision get WORSE on any of
these | "relevance" (drifts from the subsection's own topic, or adds
generic filler), "specificity" (the text itself became vaguer than
the original | not "it still lacks X"), "faithfulness" (states a
fact, number or comparison that neither the original nor the papers
above support)?
Return STRICT JSON only, no fences: {{"improved": {{"P3": true, "P7":
false}}, "worse": {{"relevance": false, "specificity": false,
"faithfulness": false}}, "note": "<= 30 words: the main reason for
any 'true' under worse, else empty"}}
```