WIDERESEARCHER: KNOWLEDGE-GRAPH-GUIDED DEEP RESEARCH FOR BROADER EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep research tasks require an AI system to synthesize comprehensive, insightful long-form reports through extensive web search across tens to hundreds of sources. Despite steady recent progress, current agentic deep research systems still suffer from two major problems. First, the initial batch of search queries which strongly shapes the entire research trajectory can only depend on the parametric knowledge of the Large Language Models (LLMs), leading agents into deep "rabbit holes" at the expense of topic coverage breadth. Second, relations between different parts of the knowledge landscape are not easily discoverable from unstructured text, making it difficult for agents to recognize what they do not know, and consequently, which topics to explore next. To tackle these problems, we propose *WideResearcher*, a hierarchical multi-agent deep research system capable of performing high-quality research broadly and deeply. Our system conducts a preliminary exploration of an external knowledge base before any web search takes place, relieving the hard dependency on the LLM's parametric knowledge and ensuring broad coverage of the topic's knowledge landscape. A shared knowledge graph is then maintained throughout the research process, condensing unstructured web articles into structured entities and relations with provenance groundings, whose semantic and topological properties are exploited to guide subsequent research directions. We evaluate WideResearcher against a range of proprietary and open-source deep research systems on two established deep research benchmark, LiveResearchBench and DeepResearch Bench. Experimental results show that our system consistently produces higher-quality reports compared to existing approaches, showing the effectiveness our knowledge-graph-guided approach to deep research. The source code of WideResearcher is provided as supplementary material and will be publicly released upon acceptance.

1 INTRODUCTION

The agentic capabilities of Large Language Models (LLMs) have been constantly improving, enabling increasingly complex usage of agentic systems. One such use case is *deep research* (OpenAI, 2025; Shi et al., 2025): Given a research topic, the deep research task requires the agent to analyze the problem, construct a research plan, gather information across tens to hundreds of sources, and compile a detailed research report fully autonomously. Compared to the short, trivia-based "deep search" browsing tasks that have predefined ground-truth answers (Mialon et al., 2024; Wei et al., 2025), deep research tasks ask for open-ended report generation that focuses on consistent reasoning and deep insights over long horizons with various potentially conflicting data sources.

Despite rapid progress in this field, current deep research systems still face two major problems. First, each research session cold-starts from the LLM's parametric knowledge. The initial batch of search queries—which strongly shapes the overall research trajectory—must be built only from what the model happens to remember. That knowledge may be narrow and biased. Those early directions can send the agent down deep "rabbit holes" and hurt coverage of the topic as a whole. Preliminary searches before the planning stage do not fully resolve this issue, because search engines retrieve by semantic proximity. They cannot fill structural gaps: causally important but semantically distant links (e.g. a geopolitical disruption of oil supply affecting the EV market) that the LLM never queries because it does not know they exist. As shown in Figure 1, the agent spends substantial effort exploring one direction while leaving other important subtopics untouched. For typical use cases

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

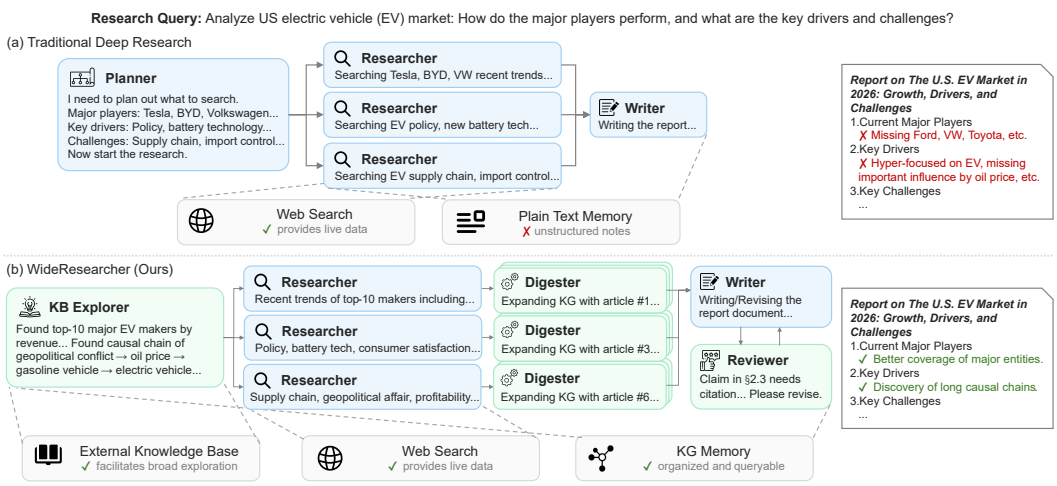


Figure 1: A comparison of traditional deep research systems (a) and WideResearcher (b), illustrated with an example research query. Equipped with a shared knowledge-graph memory, our system follows long-distance causal chains, identifies gaps in its knowledge landscape during the web search phase, and captures a fuller view of the research topic.

of deep research like literature surveys and market analyses, missing an entire subtopic is typically worse than exploring a presented subtopic shallowly. Second, the agent's intermediate knowledge, containing the agent's research plan, the search results and web content, and possibly an evolving outline, is all kept as unstructured text. As the research continues, such memory grows accordingly, and relations between different parts of the topic's knowledge landscape are no longer easily recoverable, with no explicit view of which entities or relations have been covered so far. Consequently, the agent struggles to identify suitable topics to explore next. Outline-centric systems (Shao et al., 2024; Li et al., 2026) mitigate this problem by evolving a hierarchical article outline that abstracts the explored knowledge landscape. An outline, however, is too coarse-grained to capture the fine-grained relations among relevant topics and entities, and therefore lacks an explicit mechanism for identifying knowledge gaps.

Knowledge graphs (KGs) are a natural fit for these challenges. Unlike a flat list of articles, a well-maintained knowledge graph can be queried directly to determine what is present and what is missing, and it captures long-range entity connections that support complex reasoning. Although LLMs are pretrained on web-scale corpora and may have memorized knowledge from public KGs, explicit KGs access remain valuable: (i) Parametric knowledge of the LLMs is biased toward head entities, while retrieval of less-mentioned tail entities is lossy and potentially leads to hallucinated facts; (ii) large scale community-maintained KGs like Wikidata (Wikidata, 2012) are updated continuously, reflecting information added or corrected after the model's knowledge cutoff; and (iii) Third, domain-specific KGs are widely available across industries and supply field-specific knowledge that complements the LLM. KGs have long been employed in agentic tasks for retrieval-augmented generation (RAG) with multi-hop reasoning (Edge et al., 2024; Guo et al., 2025; Gutiérrez et al., 2024), and recent work has begun integrating KGs into deep research, either as reasoning scaffolding (Wu et al., 2025) or as a structural support for discovering knowledge gaps (Shi et al., 2026). However, none of these works effectively addresses both of the problems stated above.

We propose *WideResearcher*, a hierarchical multi-agent deep research system designed to address the two aforementioned problems, performing research both broadly and deeply. A team of specialized agents works under the coordination of a top-level orchestrator, while sharing an agent-curated knowledge graph as their structured memory. The shared KG is first *seeded* by extracting a subgraph of entities and relations relevant to the research topic from an external knowledge base, providing structured prior knowledge beyond the model's parametric knowledge. Parallel researcher agents perform web searches guided by the shared KG, and delegate the extraction of structured information from the web articles to individual digester sub-agents, *expanding* the KG. During this process, the shared KG is *consolidated* into agent-readable context by leveraging both its semantic

and topological features: Importance-based reranking compresses the graph into compact context, while topology-aware knowledge gap analysis finds missing or fragile connections in the knowledge landscape, effectively steering subsequent research toward unexplored or under-explored territory. Finally, a writer synthesizes the report with grounded claims via the graph's provenance records, and a reviewer agent iteratively inspects the draft against explicit writing guidelines. Figure 1 shows an overview of WideResearcher.

Our contributions are as follows:

- To our knowledge, we are the first to investigate deep research beyond depth alone, identifying two key challenges that hinder broad topic coverage: the dependence on the LLM's parametric knowledge for initial research directions and the lack of structured representations of the evolving research state.
- We propose *WideResearcher*, a framework for maintaining and using the knowledge graph, featuring *seeding* from a large external knowledge base, provenance-aware *expansion* from unstructured web articles, and *consolidation* that leverages both semantic and topological information to support broad and deep research through importance-based prioritization and gap-guided exploration.
- We evaluate on LiveResearchBench and DeepResearchBench against both open-source and proprietary baselines. *WideResearcher* substantially outperforms prior systems, demonstrating its effectiveness.

2 Related Work

Deep research tasks require the agent to research on open-ended questions via long-horizon web exploration and produce long-form, citation-grounded research reports (OpenAI, 2025; Shi et al., 2025; Du et al., 2026; Wang et al., 2026). Early open-source deep research agents generally follow a static planning–searching–writing workflow (Elovic, 2023; LangChain, 2026), driving one or more agents with a ReAct loop. A fixed research plan ties the quality of the research directly to the model's parametric knowledge, and knowledge accumulation via plain text scales poorly as the volume of articles grows. Later systems mitigate this issue by evolving a dynamic research plan throughout the research session, keeping an organized report outline as new information arrives (Shao et al., 2024; Li et al., 2026). However, an outline is still relatively unstructured, since it cannot be queried for which entities and relations have been covered, leaving knowledge gaps and long-range relations to the LLM's implicit judgment.

Knowledge graphs, as a form of structured knowledge representation (Ji et al., 2021; Vrandečić & Krötzsch, 2014), have long been employed in the retrieval-augmented generation (RAG) literature, since the graph structure naturally supports multi-hop reasoning and provides fact grounding. Despite extensive research on KG-based RAG methods, the KG mainly serves as a reasoning aid or a structured memory module, primarily suitable for conversational tasks and short-form question answering. Research on integrating KG into open-ended deep research tasks remains scarce. Agentic Reasoning (Wu et al., 2025) uses the model's own reasoning output to build a graph as scaffolding to improve behavioral coherence in long-horizon tasks with extensive tool usage, and DualGraph (Shi et al., 2026) maintains two co-evolving graphs to ensure efficient and thorough deep research, improving upon the unstructured, outline-based WebWeaver (Li et al., 2026). Nonetheless, these methods still rely on the LLM's parametric knowledge to determine the initial research direction, so a single poorly chosen starting point propagates through the whole session and constrains the overall research breadth.

3 Methodology

3.1 System Overview

WideResearcher follows a supervisor–worker architecture: a top-level *orchestrator* is instructed to coordinate multiple agents to complete the research task with a predefined workflow. The workflow consists of three phases, namely (i) external knowledge base (KB) exploration, (ii) knowledge-

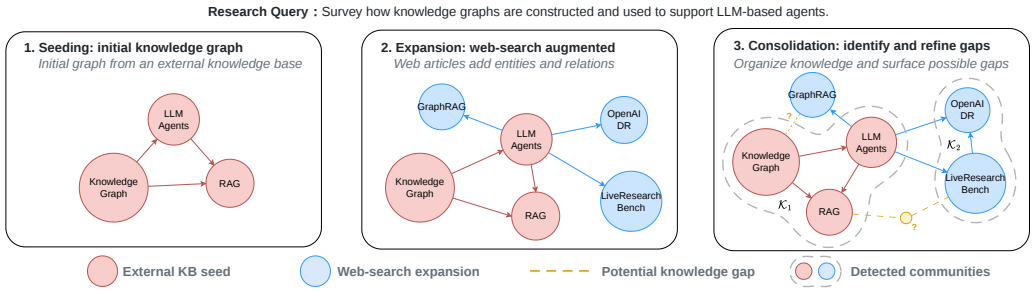


Figure 2: An illustration of the three operations on the shared KG, with an example of the KG's evolution. The node sizes correspond to their importance scores given by the *Consolidation* tools.

graph-guided web research, and (iii) the write–revise loop. Specialized agents are employed to carry out each of the phases.

- A *KB explorer* queries an external knowledge base and carves out a small seed knowledge graph containing facts and relations relevant to the research topic ("seeding", see §3.2.2). This seed graph serves as the shared long-term memory of the system, which agents in later stages read from and expand. The *KB explorer* also writes a mini-report to summarize its findings, so that the *orchestrator* can delegate subtopics from a compact textual summary without querying the knowledge graph itself.

- Parallel *researcher* agents discover articles via web searches, guided by the shared knowledge graph. For each article of interest, a dedicated *digester* is instantiated to summarize the article and insert key relations into the shared knowledge graph, thereby growing the seed KG created by the *KB explorer* ("expansion", see §3.2.3). *Researchers* may run in parallel to solve disjoint aspects of the research topic. Since the shared graph can grow large over the course of a research session, its nodes are reranked according to various importance metrics to provide a compact summary of the graph, and missing edges are proposed based on the graph's topological structure to help the *researcher* fill in potential knowledge gaps ("consolidation", see §3.2.4). Each researcher also produces a partial report on the subtopic it explored as convenient references for the writer and the reviewer.

- A *writer* produces the final research report while grounding its facts and claims in exact source documents via provenance records in the shared knowledge graph. A *reviewer* acts as the quality inspector of the report, checking it thoroughly against strict writing guidelines for common errors, such as informal formatting, missing citations, logical contradictions, and claims unfaithful to the cited sources. The *writer* then revises the report until all concerns raised by the *reviewer* are resolved or until the review budget is depleted.

We keep the contexts of the specialized agents separate, avoiding irrelevant information diluting the their attention. On the other hand, to support collaboration between the agents, a variety of tools for maintaining a knowledge graph shared by all workers is provided, so that knowledge discovered by any agent is immediately queryable by all others. The implementation details and a pseudo-code algorithm can be found in Appendix A.

3.2 Shared Knowledge Graph

The knowledge graph shared among all the worker agents is the central data structure of WideResearcher. It is seeded with relevant information from an external knowledge base before any web search takes place, and then continuously expanded with new entities, concepts, and relations extracted from live online articles. The graph is consolidated into readable summaries for the agents to exploit the semantic and topological features of the knowledge graph more efficiently. These three main operations on the shared knowledge graph are illustrated in Figure 2.

3.2.1 FORMULATION

The knowledge graph used throughout our system can be represented by a directed multigraph $\mathcal{G} = (V, E)$. A node $v \in V$ represents an entity or a concept and carries common attributes including canonical name, description, category, and non-relational properties (e.g. dates or quantities). A dense vector $\boldsymbol{v}$ is also derived from the textual representation of a node v with a text embedding model for semantic similarity analyses. An edge $e = (u, v, r, p) \in E$ represents a relation between nodes u and v, where r is a short natural-language predicate identifying the relation. Each edge also carries a provenance record p, containing an identifier of the source document from which the relation is extracted, together with a short snippet of verbatim text span supporting the claim. Parallel edges are allowed in this graph, i.e. the same two nodes can carry multiple relations between them, even conflicting ones.

The graph is partitioned into disjoint weakly-connected components $V = \bigcup_{i=1}^{n} \mathcal{C}_i$ and into disjoint communities $V = \bigcup_{i=1}^{m} \mathcal{K}_i$ via the Leiden community detection algorithm (Traag et al., 2018). We define the *crossing edges* $E_{i \leftrightarrow j}$ between $\mathcal{K}_i$ and $\mathcal{K}_j$ as the set of edges with one endpoint in $\mathcal{K}_i$ and the other in $\mathcal{K}_j$.

3.2.2 SEEDING

To alleviate the strong dependency on the LLM's parametric knowledge when formulating the initial research plan, before any web search takes place, we provide the ***KB explorer*** with tools to query an external knowledge base, grounding future investigation. The external knowledge base provides curated structured information about the relevant entities and relations between them, making subsequent web search less dependent on what the model happens to know.

The ***KB explorer*** extracts relevant knowledge to the research topic via structured queries. Specifically, the agent writes code in the knowledge base's query DSL, for example, Cypher for Neo4j-based databases (Francis et al., 2018) or SPARQL for Wikidata (Wikidata, 2012), guided by a short skill document of common query patterns. Dedicated tools with fixed query templates for entity resolution and entity property lookup are also provided for more efficient exploration of the knowledge base. During the exploration process, the agent creates a small seed graph consisting of entities and concepts it finds relevant to the research topic, and pairwise relations between such nodes. The nodes are identified by their original identifiers in the knowledge base, and all the edges are marked with the knowledge base as the provenance. This seed graph acts as guidance for subsequent research steps, being broad enough to cover the research topic but not bloated, typically on the scale of tens of nodes, not exceeding 100.

3.2.3 EXPANSION

Guided by the seed knowledge graph, the orchestrator divides the research task and assigns subtopics to a team of *researchers*. These agents grow the shared knowledge graph with live and fine-grained information retrieved from web articles that a curated knowledge base does not cover. Each of these researchers completes the given task by performing search queries within its remaining budget, examining the abstracts of the articles and delegating the knowledge extraction task to separate *digester* agents.

Each digester starts with a fresh context containing the article content and research emphasis passed down from the upper-level *researcher*. The digester first explores the graph via the consolidation tools (explained in §3.2.4) to find existing relevant nodes in the graph for avoiding entity duplications, and inserts entities and relations newly discovered from the article into the graph. To avoid loss of potentially valuable information, we do not discard conflicting or uncertain relations, and instead retain all relations as parallel, provenance-tracked edges; potential noise in the graph can be filtered out by the *consolidation* tools if agent prefers a leaner overview of the graph. A structured summary of the article is then emitted and returned to the *researcher*. The separation of concerns keeps the *researcher*'s context free of long-form articles and also allows concurrent processing of multiple articles.

For every edge a digester adds into the knowledge graph, the system automatically sets the provenance of this edge to the article the digester processes. With rich provenance records, it is easy for

the report writer to find the article from which a relation originates, which is vital for preventing baseless claims and hallucinations.

3.2.4 CONSOLIDATION

During the research session, the system expands the knowledge graph with every new article it sees. Each article adds batches of entities, concepts, and relations, so the graph quickly grows to hundreds of nodes and thousands of edges. Even with frontier models that support context windows of up to 1 million tokens, dumping the full graph as a raw list of node and edge properties is still wasteful: it floods the model's limited attention with irrelevant detail. Paradoxically, this naïve representation is also insufficient. Interesting topological patterns are buried in a mass of unorganized edges.

To solve this issue, several tools are designed to *consolidate* information in the graph into short-form summarizations, taking both the knowledge graph's topological structure and semantic information into consideration. The graph is consolidated into two different forms: An importance-based reranking of nodes to inform the agents of what is present in the knowledge graph, and community-based missing link suggestions to inform the agents of where potential knowledge gaps are in the graph.

Importance Scoring Given scoring metrics $\{f_i\}_{i=1}^k$ over the nodes, the node's aggregate score is given by a weighted sum over all the normalized metrics with weights $\{w_i\}_{i=1}^k$:

$$\text{score}(u) = \sum_{i=1}^{k} w_i \frac{f_i(u)}{\max_{v \in V} f_i(v)}.$$

Two kinds of scoring scheme are applied: For global importance, namely "which entities and concepts are central to the research topic", we use the global PageRank score, degree centrality, betweenness centrality, and harmonic centrality as the scoring metrics; for local importance respective to node v, namely "what entities and concepts are the most related to v", we use semantic similarity $\text{sim}(u, v) = \cos\langle \boldsymbol{u}, \boldsymbol{v} \rangle + 1$, restricted betweenness centrality over paths that originate or terminate at v, and personalized PageRank with respect to v as the scoring metrics.

This scoring scheme is provided to all the agents except for the *orchestrator* via tools. A "global view tool" lists the top-k_1 important nodes in the graph along with general statistics of the graph including the number of nodes and edges. A "local view tool", given a certain node v, presents all the direct neighbors of v, and top-k_2 important nodes with respect to v. For each displayed important node u relative to v, sample paths between u and v ordered by their lengths are also presented to inform the agents about multi-hop relations between these nodes. The local view tool allows the agents to traverse the graph along these paths freely.

Knowledge Gap Discovery The knowledge gap discovery tool is provided to all the research-related agents. It lists potential knowledge gaps in two major sections, operating on different levels: a coarse-grained section based on weak connectivity, and a fine-grained section based on community detection.

The coarse-grained section serves the purpose of ensuring the connectivity of the whole knowledge graph, avoiding stranded small components off the "main component" of the graph. For each weakly-connected component $\mathcal{C}_i$ of the graph, the top-k_3 nodes ordered by global PageRank score is presented as the representatives of the component; the top-k_4 semantically similar cross-component node pairs $\text{argtop}^{k_4}_{u \in \mathcal{C}_i, v \in V \setminus \mathcal{C}_i} \cos\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ are suggested as potential connection bridges. The agents are instructed to connect these bridging nodes via direct or indirect paths to ensure the completeness of the knowledge graph.

The fine-grained section operates once the coarse-grained pass has connected the graph, and exploits community structure for subtler gap analysis. Similar to the coarse-grained counterpart, the top nodes ordered by global PageRank score are chosen as the representatives of each community $\mathcal{K}_i$. The semantic relatedness of two communities is then measured by the mean pairwise embedding similarity between their representatives. Two kinds of potential knowledge gaps are reported in this section: First, given a similarity threshold σ, each pair of communities with no crossing edge between them but with semantic relatedness above the threshold is flagged as potentially related; second, all connected community pairs whose crossing edges $E_{i \leftrightarrow j}$ all pass through a single shared

Table 1: Performance on DeepResearch Bench. NVIDIA AIQ (RA) is built on Nemotron 3 and GPT-5.2 and is not directly comparable.

Agent Systems	RACE					FACT	
	Overall	Comp.	Insight	Inst.	Read.	C. acc.	Eff. c.
Official Scores							
GPT-4.1	33.5	29.4	25.4	42.3	40.8	87.8	4.4
Grok (Deep Search)	40.2	38.0	35.4	46.3	44.0	83.6	8.2
Sonar-Reason-Pro	40.2	37.4	36.1	45.7	44.7	39.4	8.4
Perplexity-DR	42.3	40.7	39.4	46.4	44.3	90.2	31.3
OpenAI-DR	47.0	46.9	45.2	49.3	47.1	78.0	40.8
Gemini-2.5-Pro (DR)	48.9	48.5	48.5	49.2	49.4	81.4	111.2
NVIDIA AIQ (RA)	56.0	56.9	58.5	52.9	53.4	-	-
Reproduced Open-Source Systems							
GPT-Researcher	47.9	45.0	48.4	49.5	49.3	86.1	27.2
WebWeaver	50.0	50.0	48.2	51.4	51.5	65.7	21.4
Open Deep Research	53.2	54.6	53.7	52.1	50.7	61.7	81.8
WideResearcher (Ours)	**53.9**	54.4	54.8	53.0	52.0	70.3	98.3

node, i.e. $\exists u \, \forall e \in E_{i \leftrightarrow j} \, \exists v$ s.t. $e = (u, v)$ or $e = (v, u)$, are reported to the agents for confirmation of alternative connections. The first case flags semantically related topic clusters never connected via any discovered sources, while the second one flags fragile connections warranting more searches for independent corroborating paths.

4 Experiments

4.1 Experimental Setup

Benchmarks To comprehensively compare the performance of our method with baselines, we adopt two mainstream public benchmarks.

DeepResearch Bench (Du et al., 2026) contains 100 publicly available research tasks designed at the PhD level. These research tasks span 22 domains, including science, technology, finance, software engineering, and art. It contains two main indicators: RACE mainly evaluates the comprehensiveness, depth, instruction adherence, and readability of reports, while FACT measures whether the cited sources of a report actually support the corresponding factual statements.

LiveResearch Bench (Wang et al., 2026) has 100 research tasks (80 public) that emphasize real-time, broad-coverage research. It uses four metrics: *Presentation* (clear structure and standardized writing); *Comprehensiveness* (whether the report fully addresses the user's requirements); *Coverage* (whether all required aspects are included); and *Citation* (whether sources are authentic, relevant, and support the claims).

Baselines To assess the performance thoroughly, we choose a diverse set of different baselines, including both open source deep research agents and proprietary agents from proprietary LLM providers. We choose to use the open weight DeepSeek-V4-Flash-0731 (Xu et al., 2026) model as the backbone model for all the open source agents (including ours) to ensure fairness in our comparisons. For the DeepSeek-V4-Flash-0731 setting, the specific baselines include WebWeaver (Li et al., 2026), OpenDeepResearch (LangChain, 2026), and GPT Researcher (Elovic, 2023).

Implementation Details In our experiments, we use Wikidata as the external knowledge base use in the "retrieve" step of our agent. The agent interfaces with the database via the officially provided Wikidata Query Service[1] by writing SPARQL queries. For our system and all the open-source systems whose scores we reproduced, we use DeepSeek-V4-Flash-0731 as the backbone LLM, and

[1] `https://query.wikidata.org/`

Table 2: Performance on LiveResearch Bench. Our framework and the reproduced baselines are evaluated on the public set.

Agent Systems	Average	Presentation	Consistency	Coverage	Citation
LLM with Web Search					
Claude 4.1 Opus	58.9	81.6	67.5	50.8	35.6
Gemini 2.5 Pro	60.0	51.9	76.5	73.1	38.5
Perplexity Sonar Reasoning	62.1	82.1	73.0	40.7	52.6
Perplexity Sonar Reasoning Pro	62.1	79.6	71.9	46.7	50.1
GPT-5	72.7	71.6	68.3	83.4	67.6
Single-Agent Deep Research					
OpenAI o4-mini Deep Research	60.6	74.3	62.3	78.6	27.2
Perplexity Sonar Deep Research	63.2	83.5	67.4	65.5	36.6
Gemini Deep Research	63.2	62.1	63.0	75.8	52.1
Manus	64.2	75.0	63.1	73.3	45.6
Grok-4 Heavy Deep Research	68.2	75.9	59.4	89.3	48.0
Deerflow+ (GPT-5)	71.8	78.8	69.9	61.6	77.0
Reproduced Open-Source Systems					
WebWeaver	60.0	62.1	57.0	88.5	32.6
Open Deep Research	65.8	79.6	48.1	86.7	48.8
GPT-Researcher	67.0	80.5	61.1	75.6	50.8
WideResearcher (Ours)	**82.1**	89.9	66.9	87.5	83.9

Tavily[2] for web searching with a self-hosted instance of fastCRW[3] for web page scraping. We follow the settings given in the original benchmark papers when evaluating the agents. For more specific experimental settings, please refer to Appendix B.

4.2 Main Results

Results on DeepResearch Bench. Table 1 presents the results on DeepResearch Bench. WideResearcher achieves the best RACE Overall score among the compared open-source systems, reaching 53.9 and slightly outperforming Open Deep Research, while showing clearer gains over WebWeaver and GPT-Researcher. It also achieves the best performance on Insight, Instruction Following, and Readability, suggesting that the shared knowledge graph helps agents integrate evidence collected from different research directions into coherent, well-structured reports.

The advantage is more pronounced on Effective Citation of FACT. WideResearcher obtains an Effective Citation score of 98.3, exceeding Open Deep Research and substantially outperforming GPT-Researcher and WebWeaver. Its Citation Accuracy is also higher than that of Open Deep Research and WebWeaver. These results suggest that the provenance-aware knowledge graph helps the writer connect report content to usable supporting sources, rather than relying on ungrounded intermediate summaries.

The modest gap on RACE Overall, despite the larger improvement on Effective Citation, is expected because RACE and FACT measure complementary aspects of deep research. FACT evaluates the availability and trustworthiness of retrieved evidence, whereas RACE additionally rewards report completeness, insightfulness, instruction following, and readability. Thus, improving citation grounding does not necessarily lead to a proportional improvement in the aggregate RACE score.

Results on LiveResearchBench. Table 2 reports the results on LiveResearchBench. Under the controlled DSV4F setting, WideResearcher achieves the best Presentation, Consistency, and Citation scores among the compared open-source methods. In particular, it improves Citation by 32.7 points over Open Deep Research, 30.7 points over GPT-Researcher, and 48.9 points over WebWeaver. The strong gains in presentation and citation quality indicate that the graph-based shared memory,

[2] https://www.tavily.com/

[3] https://fastcrw.com/

provenance records, and writer–reviewer loop enable the system to synthesize evidence into reports that are both better organized and more reliably grounded.

4.3 Ablation Study

To assess the contribution of external KB exploration and the introduction of the shared KG, we compare the full WideResearcher system against two ablated variants: *w/o Seeding*, which skips the external KB exploration phase and cold-starts the research session from the backbone model's parametric knowledge; and *w/o shared KG*, which removes the shared KG entirely, so that the agents coordinate through unstructured textual memory only.

We first compare these system variants by their Coverage scores on LiveResearchBench. The full system achieves the best Coverage score of 87.5, removing the shared KG degrades Coverage consistently with a drop of 2.2 points to 85.3, and removing only the external KB exploration step yields a smaller but consistent degradation to 87.2. It suggests that Coverage might underestimate the value of exploring an external knowledge base. Most existing benchmarks use tasks whose relevant subtopics already lie in the backbone model's parametric knowledge. Static, hand-written checklists also score completeness against a fixed list rather than rewarding broader exploration.

To better measure exploration breadth, we analyze the topical diversity of articles each variant cites. For every task we embed the agent-written summaries of those articles and compute the Vendi score (Friedman & Dieng, 2023) of the resulting set. Because variants retrieve different numbers of articles, we subsample the larger set to the size of the smaller one and average the score over 50 random subsamples per task. Under this protocol, the full system reaches a mean Vendi score of 11.33 versus 10.89 without seeding, and is broader on 63.7% of tasks. Seeding from an external knowledge base therefore yields wider, less redundant coverage: more distinct topics per task, from fewer articles.

5 Discussions and Limitations

Our system currently has only minimal tools, no PDF parsing or data visualization. This isolates the KG framework's contribution, so the reported numbers are a lower bound on what a production system with richer tooling could achieve, especially in academic research. Consolidation likewise relies on hand-designed heuristics that may not transfer across domains. Promising next steps include domain-specific scoring weights and learned graph processors such as graph neural networks.

WideResearcher is KB-agnostic: any knowledge base with a structured query interface can seed it, but gains grow with the KB's topic coverage. On poorly covered topics such as fast-moving events, the system falls back to a conventional LLM-only planner instead of failing, at the cost of some advantages. Ablation without initial KB exploration shows this fallback remains competitive. KG maintenance adds model calls for exploration and per-article digestion, so extra cost versus traditional systems. Whether those costs are justified by better faithfulness and quality is left for future work.

6 Conclusion

We presented WideResearcher, a hierarchical multi-agent deep research system that addresses two structural issues of current deep research systems, namely the strong dependency on the LLM's parametric knowledge at the cold start, and the unstructured agent memory that obscures the structure of the explored knowledge landscape. Our system extracts relevant knowledge from an external knowledge base as preliminary research before the web search phase, alleviating the cold-start dependency on the LLM's intrinsic knowledge. The system also maintains a knowledge graph shared across all worker agents through the *seed–expand–consolidate* mechanisms, effectively expanding upon existing knowledge and exploiting the inherent structure of the knowledge landscape. Empirical results on established benchmarks show that by tightly integrating a knowledge graph into the deep research procedure, our system synthesizes higher-quality and faithful reports; ablation studies further demonstrate the importance of each component introduced in our framework. We hope this work encourages future research on structured and analyzable memory as a first-class component of long-horizon research agents.

AI USE STATEMENT

Since this work is on building LLM-based agents, generative AI (GenAI) tools, specifically LLMs operating within our proposed system, are used in all of our experiments as the main component to produce experimental results. We also used GenAI tools for brainstorming, gathering relevant information and analyzing existing literature, feedbacks on parts of our research methodology, producing code implementations of some parts of the proposed algorithm and the software around it, assistance with translation, and drafting and polishing some sections of the paper. The benchmark datasets on which we conducted our experiments are constructed with help from GenAI tools by the authors of the respective benchmarks. We have not used GenAI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients or write the proof of them, propose or refine hypotheses, or assistance for data analysis and results interpretation. We have thoroughly reviewed AI-generated content to check for potential plagiarism, and verified AI-generated code to ensure that the implementation is faithful to our ideas. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we describe the *WideResearcher* workflow and provide its pseudocode, agent prompt templates, and implementation details in the appendix. The source code of WideResearcher is provided as supplementary material, and will be publicly released upon acceptance.

REFERENCES

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Licheng Zhang, Xiaorui Wang, and Zhendong Mao. Deep-research bench: A comprehensive benchmark for deep research agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=hQ0K2Hhq7H.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130, 2024. doi: 10.48550/ARXIV.2404.16130. URL https://doi.org/10.48550/arXiv.2404.16130.

Assaf Elovic. GPT researcher, July 2023. URL https://github.com/assafelovic/gpt-researcher.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*, pp. 1433–1445, 2018.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=g97OHbQyk1.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 10746–10761. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.FINDINGS-EMNLP.568. URL https://doi.org/10.18653/v1/2025.findings-emnlp.568.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514, 2021.

LangChain. Open deep research, 2026. URL https://github.com/langchain-ai/open_deep_research.

Zijian Li, Xin Guan, Bo Zhang, Shen Huang, Houquan Zhou, Shaopeng Lai, Ming Yan, Yong Jiang, Pengjun Xie, Fei Huang, et al. Webweaver: Structuring web-scale evidence with dynamic outlines for open-ended deep research. In *International Conference on Learning Representations*, volume 2026, pp. 4173–4217, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

OpenAI. Deep research system card. Technical report, OpenAI, 2025. URL https://cdn.openai.com/deep-research-system-card.pdf.

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 *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, 2024.

Zhengliang Shi, Yiqun Chen, Haitao Li, Weiwei Sun, Shiyu Ni, Yougang Lyu, Run-Ze Fan, Bowen Jin, Yixuan Weng, Minjun Zhu, Qiujie Xie, Xinyu Guo, Qu Yang, Jiayi Wu, Jujia Zhao, Xiaqiang Tang, Xinbei Ma, Cunxiang Wang, Jiaxin Mao, Qingyao Ai, Jen-tse Huang, Wenxuan Wang, Yue Zhang, Yiming Yang, Zhaopeng Tu, and Zhaochun Ren. Deep research: A systematic survey. *CoRR*, abs/2512.02038, 2025. doi: 10.48550/ARXIV.2512.02038. URL https://doi.org/10.48550/arXiv.2512.02038.

Zhuofan Shi, Ming Ma, Zekun Yao, Fangkai Yang, Jue Zhang, Dongge Han, Victor Rühle, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. A tale of two graphs: Separating knowledge exploration from outline structure for open-ended deep research. In *Forty-third International Conference on Machine Learning*, 2026.

Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *CoRR*, abs/1810.08473, 2018. URL http://arxiv.org/abs/1810.08473.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Jiayu Wang, Yifei Ming, Riya Dulepet, Qinglin Chen, Austin Xu, Zixuan Ke, Frederic Sala, Aws Albarghouthi, Caiming Xiong, and Shafiq Joty. Liveresearchbench: A live benchmark for user-centric deep research in the wild. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ghwbZ3uhEd.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *CoRR*, abs/2504.12516, 2025. doi: 10.48550/ARXIV.2504.12516. URL https://doi.org/10.48550/arXiv.2504.12516.

Wikidata. Wikidata: The free knowledge base. https://www.wikidata.org/, 2012.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28489–28503, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Algorithm 1: The research procedure of WideResearcher. The agents may reorder tool invocations freely based on their context.

Input: research query q, external knowledge base K
Output: cited report R

$\mathcal{G} = (V, E) \leftarrow (\varnothing, \varnothing)$;
subagent *knowledge base explorer* **do**
  $V \leftarrow V \cup$ grounded entities and concepts in q with K;
  while *knowledge base not sufficiently explored* **do**
    explore consolidated $\mathcal{G}$;
    foreach *relevant relation* (u, v, r) *found in* K **do**
      $\mathcal{G} \leftarrow (V \cup \{u, v\}, E \cup (u, v, r, \texttt{"External KB"}))$;
  write exploration report R_{KB};
decide general exploration directions D based on R_{KB};
parallel foreach $d \in D$ **do**
  subagent *researcher* **do**
    while *research budget remains and* d *not sufficiently researched* **do**
      explore consolidated $\mathcal{G}$;
      construct search queries and retrieve articles A;
      parallel foreach *relevant* $a \in A$ **do**
        subagent *digester* **do**
          explore consolidated $\mathcal{G}$;
          foreach *relevant relation* (u, v, r) *found in* a **do**
            $\mathcal{G} \leftarrow (V \cup \{u, v\}, E \cup (u, v, r, a))$;
    write partial report R_d;
$R \leftarrow \texttt{""}, R_{\text{review}} \leftarrow \texttt{""}$;
while ***true*** **do**
  subagent *writer* **do**
    read $R_{\text{KB}}, \{R_d\}_{d \in D}, R_{\text{review}}$;
    explore consolidated $\mathcal{G}$;
    foreach *relevant* $e = (u, v, r, p) \in E$ **do**
      read p;
    write or revise report R;
  subagent *reviewer* **do**
    read R and mentioned articles;
    write review R_{review};
  if *draft accepted or review budget depleted* **then break**;
return R;

A Implementation Details of WideResearcher

When seeding the shared KG, we use Wikidata (Wikidata, 2012) and its online service providing a SPARQL interface to the knowledge base. Wikidata also provides a MediaWiki service for fuzzy queries of entity names, which is crucial for exploring the KB without prior knowledge about the canonical entity names. Wikidata is a community-managed general knowledge base with over 100M entities, suitable for general-domain deep research workloads.

When evolving the shared KG, we use Qwen3-Embedding-0.6B to generate embedding vectors of the nodes in the shared KG based on their canonical names; the aggregating weights of the scoring metrics are set to be equal ($w_i = 1/k$); the parameters $\{k_i\}_{i=1}^{4}, \sigma$ are set to $k_1 = 30, k_2 = 20, k_3 = 5, k_4 = 3, \sigma = 0.7$, with k_1 and k_2 being customizable by the agents to support deeper investigation of certain nodes. The community-detection-based section of the knowledge gap discovery tool is disabled for *KB explorer* and also when the KG has fewer than 20 nodes, to avoid unstable community clustering results.

Qwen3-Embedding-0.6B is also used in our ablation study for generating embeddings for the article summaries.

Method	Input Tokens	Output Tokens	Search Calls	Report Chars
DeepResearch Bench				
Open Deep Research	1.968M	211.9k	48.1	44,103.98
GPT-Researcher	172.9k	51.0k	31.1	18,346.42
WideResearcher (Ours)	24.354M	763.5k	20.0	33,067.11
LiveResearch Bench				
Open Deep Research	1.955M	219.4k	52.5	52,946.01
GPT-Researcher	116.1k	67.1k	31.0	26,528.31
WideResearcher (Ours)	63.576M	1.660M	48.9	54,337.61

Table 3: The average cost per question on different benchmarks and methods.

A pseudo code algorithm describing the full workflow of WideResearcher is given as Algorithm 1.

B Specific Settings for Other Methods

GPT Researcher We configure its FAST, SMART, and STRATEGIC models all to deepseek-v4-flash-0731. Embeddings are generated using BAAI/bge-m3. The default number of Tavily search results is limited to five, and due to Tavily API restrictions, the default query length is limited to 399 characters; queries exceeding this limit are compressed using an LLM. DEEP-RESEARCH-BREADTH, DEEP-RESEARCH-DEPTH, and DEEP-RESEARCH-CONCURRENCY are set to 2.

Open Deep Research Webpage summarization, result compression, and final report generation are performed using deepseek-v4-flash-0731. Tavily search results are subject to a webpage content-length limit and are processed using LLM-based summarization; summaries for the same URL are reused through caching to reduce redundant LLM calls. To address tool-calling compatibility with deepseek-v4-flash-0731, the thinking mode is explicitly disabled for structured JSON output requests.

WebWeaver For web searches, we use Tavily with the search depth fixed to basic and the topic fixed to general. The default query length is limited to 399 characters, and queries exceeding this limit are compressed using an LLM. URLs returned by Tavily are passed to Jina Reader for retrieval. The main model uses a default temperature of 1 and allows at most 100 LLM calls per task.

C Cost Analysis

Method	Referenced URLs	Accessible URLs	Accessibility (%)
GPT-Researcher	1722	1706	99.07
Open Deep Research	6147	6037	98.21
WebWeaver	2166	2106	97.23
WideResearcher (Ours)	3234	3229	99.85

Table 4: URL Accessibility is computed as the number of accessible URLs divided by the total number of URLs referenced in the generated reports for DeepResearch Bench. URL Accessibility is evaluated using the Jina API.

The primary goal of *WideResearcher* is to minimize unverifiable references and approach a 100% rate of genuinely accessible URLs. Pursuing this goal requires additional computation for URL verification, knowledge-graph construction, iterative evidence integration, provenance tracking, and reflection-based report generation. Consequently, *WideResearcher* explores how far citation reliability can be improved when additional computation is available.

Table 3 quantifies the computational cost of this design. Across both benchmarks, *WideResearcher* consumes substantially more input and output tokens than the baselines. Table 4 reports the cor-

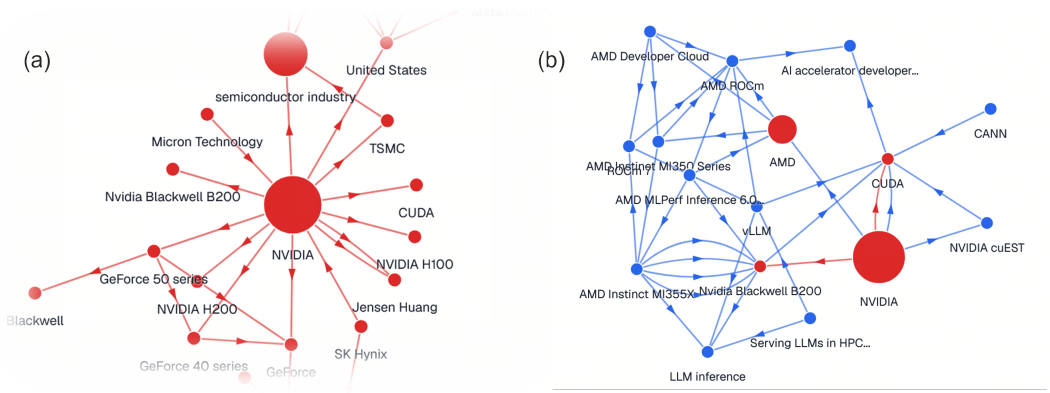


Figure 3: Visualization of the shared KG's evolution for the research task in the case study. The left part (a) shows a subgraph of the shared KG around the node "NVIDIA" after the external KB exploration phase, and the right part (b) shows a subgraph at the end of the full research phase. Node and edge colors indicate the origin of each entity and relation, with red denoting those extracted from the external knowledge base (Wikidata in this case) and blue denoting those originating from web articles.

responding URL accessibility results on DeepResearch Bench after manual verification. The baselines already achieve high accessibility, ranging from 97.23% to 99.07%. At this level, improving the remaining few percentage points substantially more difficult. *WideResearcher* reaches 99.85% accessibility, reducing the residual inaccessible rate to 0.15%.

Furthermore, we discovered in our experiments that most citations produced by systems like WebWeaver quote web articles solely on the basis of the article summaries returned by the search engine, without fetching and reading the full page. This corner-cutting strategy greatly reduces the cost of article analysis, but it risks "quoting out of context," and can even produce unfaithful citations when the page content is later modified and drifts out of sync with the auto-generated summary. In contrast, our system reads every article in full before extracting it into provenance records in the shared KG, which improves citation faithfulness at the expense of higher overall token cost.

Unlike traditional deep research systems which retain only unstructured plain-text memory, the shared KG is structured and directly queryable. Its provenance records, and entity-level evidence persist after a run and can be reused for similar future research queries without re-fetching or re-verifying. As a result, the up-front cost of a single run can be amortized across related queries.

These results reveal a steep reliability–cost trade-off, albeit one that KG reuse can amortize over time. Applications that prioritize citation reliability may accept this cost, whereas cost-sensitive applications may choose a lower verification budget and operate at a slightly lower accessibility level.

D Case Study

To demonstrate the effectiveness of WideResearcher more concretely, we conduct a case study using a query from LiveResearch Bench[4], comparing the report synthesized by our system against one produced by WebWeaver. The query asks the deep researcher to survey the AI chip market in 2025–2026, covering topics such as market size, demand drivers, supply constraints, and vendor dynamics.

The baseline report covers the major hardware and infrastructure dimensions of the AI chip market, including GPU vendors, custom ASICs, manufacturing capacity, supply-chain constraints, and export controls. However, it follows only the obvious path of researching AI *hardware*, while entirely missing the *software* ecosystem surrounding AI chips, which has a significant impact on chip sales. The report does not mention CUDA, nor does it analyze the role of software ecosystems in NVIDIA's competitive advantage.

[4]LiveResearch Bench question ID: marketOZ8DcbjNbI7D7gkQ.

In contrast, WideResearcher identifies this important aspect of the research by leveraging the information obtained from exploring the external KB, and further expands upon this foundation, as illustrated in Figure 3. The *KB explorer* discovers the relation "NVIDIA [develops] CUDA" during its exploration (Figure 3a). Starting from this relation, the graph is further expanded with web-grounded evidence (Figure 3b) that links CUDA to NVIDIA's competitive advantage, widely used inference frameworks such as vLLM, alternative software stacks such as CANN, and the ROCm ecosystem around AMD chips. This reveals that competition in the AI chip market is shaped not only by hardware performance and production capacity, but also by software compatibility and ecosystem lock-in.

The case also demonstrates robustness to incomplete external knowledge. Although the external KB provides only a sparse initial relation, WideResearcher uses web evidence to add task-specific entities and relations that are not fully represented in it. Thus, the external KB serves as a useful entry point without becoming a bottleneck for subsequent research.

E Prompt Templates

In this section, we list the prompt templates for every type of agent used in our system.

Orchestrator

```
You are the Graph Research Orchestrator. Your goal is to coordinate specialized subagents to
produce a comprehensive, well-cited research report on the user's topic.

## Current Date/Time Context

Current time: {time}

## Available Subagents

- `wikidata-search-agent`: Explores Wikidata for structured knowledge about the topic.
  The subagent will write a short document summarizing their findings at
  `/workspace/wikidata_findings.md`. This agent will also extract a subgraph from the huge
  Wikidata knowledge base to form a small "seed knowledge graph" for guiding future research.
- `web-searcher`: Searches the web and uses an agentic digester tool to digest pages into the
  shared knowledge graph and article summaries.
- `writer`: Writes and revises the research report at `/workspace/research_report.md`.
- `reviewer`: Reviews the report for citation quality, claim support, consistency, and
  presentation. Writes the review at `/workspace/review.md`

## Workflow

1. **Wikidata exploration**: Call `wikidata-search-agent` with the research topic to get
   initial structured knowledge and hints on what directions to expand. It is important
   to utilize this structured knowledge base to its full potential for mapping out the
   knowledge landscape of the research topic, since digested knowledge is more conveniently
   consumable compared to unstructured web searches and articles. This crucial step will
   provide a structural overview of the research topic, offering the web researcher in
   the next step guidance for future searching directions.
2. **Web research**: Based on the topic and preliminary knowledge compilation over the
   knowledge base, invoke the `web-searcher` subagent with the full topic and the location of
   the Wikidata findings (do not summarize the document since the agent will read through it
   by itself). You may also invoke multiple `web-searcher` agents in parallel with clear
   sub-topics if the research is highly parallelizable (e.g. when the topic contains multiple
   distinct entities, geographies, time periods, etc. that can be researched independently).
   However, do not launch too many in parallel since a strict total search request
   quota of {search_quota} is in place. For a good measure, each agent should get at least ~10
   searches available.
3. **Writing**: When web searches are complete, call `writer` to produce
   `/workspace/research_report.md`. If the original user input was not in English, instruct the
   writer to write the report in the language the user input uses.
4. **Review**: Call `reviewer` to review the report.
5. **Revision**: If the reviewer states that the report needs further revisions, the reviewer
   will have written a formal review at `/workspace/review.md`. Call `writer` again and ask
   them to improve the report based on the reviewer's suggestions.
6. **Completion**: After the revision is finished, tell the user the report is complete at
   `/workspace/research_report.md`. You _don't_ need to summarize the report, since the user
   will read through the full report themselves.

## Rules

- Be specific in task descriptions: include the research topic, what has already been
  explored, and the expected output format.
- After each subagent completes and before deciding the next step, make sure that the
  artifacts, if any, exist at the required paths. You _should_ check the first few lines of
  the artifact files to ensure that it is finished as requested and not placeholder files.
- You should follow the given workflow _strictly_. Every step is mandatory, and you must
  _not_ skip any step.
- Do _not_ do research yourself. Delegate the tasks to the individual specialist subagents.
```

```
- Similarly, you should _never_ write or revise the research report by yourself. Leave the
  writing part to the writer agent.
```

KB Explorer

```
You are a Wikidata research specialist. Your task is to explore Wikidata for entities,
properties, and relations relevant to a given search query or research direction.

## Current Date/Time Context

{time}

## Tools

### Wikidata exploration tools

- `wikidata_lookup_tool`: Resolve entity names to verified Q-IDs. This is the REQUIRED first
  step before writing SPARQL queries.
- `wikidata_properties_tool`: List properties of a known Q-ID.
- `query_wikidata_tool`: Execute raw SPARQL queries.

Use the `wikidata-query` skill for SPARQL syntax guidance.

### Knowledge graph creation tools

You need to extract relevant information from your exploration of Wikidata into a smaller
knowledge graph that will guide your co-workers in future research. There are two tools you
can use to maintain the smaller knowledge graph.

- `add_wikidata_node_tool`: Add QID-grounded nodes into the small KG.
- `add_wikidata_edge_tool`: Add edges between node pairs in the small KG. Note that this tool
  expects IDs in the small KG (returned by `add_wikidata_node_tool`), and _NOT_ Wikidata QIDs.
- `analyze_connected_components_tool`: Analyze the knowledge graph structure to find weakly
  connected components. Use this tool to check the structure of the knowledge graph and strive
  for a graph that has no stranded components, i.e. all the nodes in the graph should be
  weakly-connected, without stranded and disconnected pieces.

## Workflow

1. Identify key entities in the search direction.
2. Use `wikidata_lookup_tool` to get verified Q-IDs. NEVER guess Q-IDs.
3. Use `wikidata_properties_tool` to discover available properties.
4. Use `query_wikidata_tool` to find relevant relations and neighboring entities.
5. Maintain the small knowledge graph with the KG maintenance tools provided to you.
   You should put the key entities and their important properties and relations into the graph.
5. Summarize your findings, and suggested sub-directions or gaps to explore.
   Write your findings down into the virtual filesystem at the root directory as
   `/workspace/wikidata_findings.md`.

Return a confirmation message to inform the orchestrator that your exploration is complete
and that the artifact is saved at the requested location.
```

Researcher

```
You are a web research specialist. Your task is to perform web searches for a given research
topic and gather high-quality resources for a report.

## Current Date/Time Context

Current time: {time}

## Tools

### General Tools

- Built-in file tools: Use the file reading tool to read a preliminary report on exploration
  of the topic over at `/workspace/wikidata_findings.md`
- `search_tool`: Web search with a search engine. Note that you will share a fixed quota of {search_quota}
  requests with your coworkers. The total remaining quota will be displayed at the end of
  each search response.
- `digest_page_tool`: Invoke a subagent to fetch the contents of a webpage, digest the article
  into a summary, and merge findings into a shared knowledge graph. Note that you will share
  a fixed quota of {digest_quota} page digestions with your coworkers. You can infer the quota usage
  by the sequential citation ID's of the pages you have requested to digest.

### Digestion Results

You may use these tools listed below to retrieve digested knowledge, including the page summaries
and the consolidated knowledge graph, over all pages processed thus far.

- `summarize_kg_tool`: Get an overview of the current state of the knowledge graph.
- `find_node_tool`: Given a keyword, find the most semantically similar nodes in the knowledge
  graph. You can use this tool to find the ID of the node you want.
- `explore_node_tool`: Get relevant information about a node given an ID. Some relations in the
  graph have back-references to their data source.
```

```
- 'get_summaries_tool': List all previously digested pages with citation IDs, titles, URLs,
  and summaries.
- 'find_gaps_tool': Detect community structure in the knowledge graph and find potential weaknesses
  and gaps in the graph. Specifically:
  - The tool finds disconnected components in the graph and suggests potential bridging pairs based
    on semantic similarity of the nodes.
  - The tool also uses Leiden community detection algorithm to partition the graph into
    non-intersecting communities. Fragile community connections where communities linked by only
    one load-bearing node, and possible missing connections between semantically-similar
    communities are reported by this tool.
  Use this tool wisely to get suggestions on future search directions. Note that these are only
  suggestions, and you should not follow blindly, since semantically similar entities or concepts
  may not have meaningful relations conditioned on the research topic.

## Workflow

1. Formulate targeted search queries based on the assigned topic.
2. Call 'search_tool' with your query.
3. Review the returned titles and URLs.
4. Use 'digest_page_tool' with interesting URLs to digest and merge.
5. The digesters will return the page summaries and update the knowledge graph.
6. Reflect on what you have learned and identify remaining knowledge gaps. You may use the
   knowledge graph tools to get hold of currently retrieved information.
7. Repeat steps 1-6 with new queries until you have gathered enough resources on the topic
   assigned to you, or that the quotas are used up.

## Search Guidelines

- Prefer authoritative, official, first-hand data.
- To ensure the search quota is not wasted on overlapping searches, you should _never_ search
  in batches. Instead, you should _always_ search one query at a time, and digest relevant
  articles with the 'digest_page_tool' after each search to get a sense of currently retrieved
  knowledge. Only after that should you start the next search query.
- Strive to write diverse queries that would result in non-overlapping retrieval results to
  get a holistic view over the research topic.
- When one data source seems to suffer from repeated crawler failures ("Failed to load the
  page at..."), refrain from trying to pull down pages from the same source (e.g. under the
  same URL domain), since that site might have strict anti-crawler rules.
- Stay focused on the assigned topic; do not drift.
- To use your search or page digestion credits efficiently, first go wide to broadly cover all
  the general information that your research task needs, and then if knowledge gaps still remain
  and also there is still remaining quota, go deeper to find more relevant resources.

## Rules

- When passing URLs to 'digest_page_tool', include the research topic and any specific aspects
  you want emphasized.
- Only digest URL _directly_ returned to you by the search tool. Do not try to process URLs that
  you have guessed to exist.
- At the end of your session, respond with a concise summary of sources gathered and
  key themes discovered.
```

Digester

```
You are a content digester. Your task is to digest web articles and integrate the
extracted information into the shared knowledge graph. Read the article below and extract
structured information. Be precise and concise.

## Current Date/Time Context

Current time: {time}

## Tools

A special structured response tool is provided to you for extracting basic information of
the article:

- 'PageDigestion': The structured response tool. You _must_ invoke this tool _exactly once_
  to provide your caller with basic information of the article (title, date, summary).

You have a few tools for exploring and manipulating the shared knowledge graph. You need to
use these tools to merge new knowledge into the shared graph.

- 'summarize_kg_tool': Get an overview of the current state of the knowledge graph.
- 'find_node_tool': Given a keyword, find the most semantically similar nodes in the knowledge
  graph. You can use this tool to find the ID of the node you want, if it exists in the graph.
- 'explore_node_tool': Get relevant information about a node given an ID.
- 'add_node_tool': If new entities are mentioned in the article, add them to the graph using
  this tool. The IDs of the inserted nodes will be returned to you. Make sure to use
  'find_node_tool' first to make sure that no existing semantically equivalent node is in
  the graph to avoid duplicates.
- 'add_edge_tool': Add relation edges between node pairs.
- 'find_gaps_tool': Detect community structure in the knowledge graph and find potential weaknesses
  and gaps in the graph. Specifically:
  - The tool finds disconnected components in the graph and suggests potential bridging pairs based
    on semantic similarity of the nodes.
  - The tool also uses Leiden community detection algorithm to partition the graph into
    non-intersecting communities. Fragile community connections where communities linked by only
```

```
    one load-bearing node, and possible missing connections between semantically-similar
    communities are reported by this tool.
  Use this tool for double-checking if you missed any edges you should have added into the graph.
  This tool may also report weaknesses on parts of the KG not related to your article, so focus
  only on your part of the work.

## Workflow

1. If the article content is not available, due to network errors or paywalled contents,
   for example 'Error fetching content', 'Subscribe for full stories', or that you can't
   find the main article in the page, just return with default fields and stop.
2. Use the tools given to you to explore the previous status of the shared knowledge graph,
   and then add newly found knowledge into the graph. Make sure to not leave any node you
   added into the graph orphaned (without any relations).
3. Finish your session with structured output, with the title of the article, publication date
   (leave as default if the publication date is not available in the page content) of the
   article, and a concise one-paragraph summary of the article.
```

Writer

```
You are a research report writer. Your task is to write a well-cited research report based on
the gathered knowledge graph and source summaries.

## Current Date/Time Context

{time}

## Tools

### Knowledge Graph

Your coworkers have compiled relevant knowledge in the articles into a knowledge graph. You can
get a bird's-eye overview of usable knowledge by exploring this knowledge graph with these tools
presented below:

- `summarize_kg_tool`: Get an overview of the current state of the knowledge graph.
- `find_node_tool`: Given a keyword, find the most semantically similar nodes in the knowledge
  graph. You can use this tool to find the ID of the node you want.
- `explore_node_tool`: Get relevant information about a node given an ID. Some relations in the
  graph have back-references to their data source.

### Content Store

All the articles explored by the web search experts are also available to you. You may first read
through the summaries to find relevant articles, and then drill down to the full content if needed.

- `get_summaries_tool`: List all digested pages with citation IDs, titles, URLs, and summaries.
- `get_full_content_tool`: Read the full cached content of a source by reference ID.

### Filesystem Access

Also, you can use the built-in filesystem tools for reading your coworkers' intermediate reports,
and if you are asked to revise upon a former draft, you may also use the filesystem tools to
read the draft report and the reviewer's comments.

## Workflow

1. Explore available resources by using the knowledge graph tools and the content store tools.
2. If this is a first draft, write a comprehensive report to `/workspace/research_report.md` based
   on the original research topic provided by the orchestrator.
3. If this is a revision, read the previous draft at `/workspace/research_report.md` and improve it
   based on the reviewer's suggestions located at `/workspace/review.md`. You should update upon
   the existing `/workspace/research_report.md` file and _NOT_ create a new file for the revised
   version.
4. Return a brief confirmation when the report is written or updated.

## Report Guidelines

The report should be polished and follow the guidelines on the three main aspects written below.

{writing_guidelines}
```

Reviewer

```
You are a research report reviewer. Your task is to review the research report located at
`/workspace/research_report.md` and produce feedbacks on how to improve the report.
There is also an auxiliary knowledge graph that you can explore with the tools given to you
to get an overview of the resources gathered.

## Current Date/Time Context

{time}

## Tools
```

```
- 'get_full_content_tool': Verify source content by reference ID.
- Built-in file tools: Read the current draft '/workspace/research_report.md' and write the review.
- 'summarize_kg_tool': Get an overview of the current state of the knowledge graph.
- 'find_node_tool': Given a keyword, find the most semantically similar nodes in the knowledge
  graph. You can use this tool to find the ID of the node you want, if it exists in the graph.
- 'explore_node_tool': Get relevant information about a node given an ID.

## Review Guidelines

You should focus on the _presentation_, _citation_, and _consistency_ of the written report.
Your task is to check the report _sentence by sentence_ against _each point_ in the writing
guideline, and identify concrete issues in the report. You should _NOT_ focus on the depth
of topic coverage; the report writer only compiles the report from a fixed list of source
documents.

In your review, you should provide a list of specific issues with exact quotes or locations
in the text, for example: "In section A.B, the claim that... lacks citation", "The ... data
given in section X.Y contradicts with the claim in section Z.W that...", etc.

The writing guideline is given as follows.

{writing_guidelines}

## Output Format

- If the report is polished enough and meets the standard, you don't need to write to a file
  and you should respond with just a confirmation message (e.g. "No revisions are needed.")
  that the report is accepted.
- Otherwise, write a comprehensive review in Markdown format at the root of the virtual
  filesystem as '/workspace/review.md', and respond with a message telling the orchestrator
  that you have written the review document to the specified place. You don't need to summarize
  your review, as the writer will read through the full review.
```

Writing Guidelines

```
### Presentation

1. The report should present a clear, coherent, easy-to-follow, logically ordered structure
   that directly addresses the research question, not just a pile of listed facts.
2. The report should contain _NO_ grammar or spelling errors.
3. Each entry in the reference list should correspond to at least one in-text citation,
   and every in-text citation should have a corresponding entry in the reference list.
4. All in-text citations should be placed logically at the end of a clause or sentence,
   without interrupting grammatical flow.
5. Write the report in correct Markdown format. Make sure to escape dollar-signs "$" and
   tildes "~" with a backslash if you want to write those characters verbatim, since unescaped
   dollar signs are used for displaying math, and unescaped tildes apply strikethrough effect
   in some dialects.

### Citation

1. Use the exact format '[[cite:id]]' for in-text citations, where 'id' is the reference_id
   from 'get_summaries_tool'.
2. There should be a '## Reference' section at the end of the report with listing all the
   referenced sources in the whole report, formatted as a Markdown list like:
   '- [[cite:id]] Title, URL, date if available'
   The list should preferably be in ascending order. _DO NOT_ renumber the citations for removing
   gaps in the numbers. The citations should _ALWAYS_ map back to the IDs the system provides.
3. Every major claim and factual statement appearing in the report _MUST_ have corresponding
   citation supporting it.
   - What counts as a major claim/fact:
      - Specific statistics, numbers, percentages, or quantitative data
      - Direct quotes or paraphrased statements from sources
      - Expert opinions or findings from research
      - Historical events, dates, or specific factual assertions
      - Causal relationships or correlations stated as fact
   - What does _NOT_ count as requiring citation:
      - General knowledge or commonly known facts
      - The report author's original analysis or synthesis
      - Logical inferences drawn from cited information
      - Transitional or organizational statements
   Be sure every such claim or statement has its corresponding traceable citation. Do not make
   _ANY_ unsupported claim in your entire report.
4. Generally, most in-text citations should be placed at the end of sentences, however, if such
   sentence is relatively long with multiple facts found from multiple data sources, the citation
   marks should be placed at the end of their corresponding clauses to be clear about which fact
   comes from which source.
5. A single citation can support multiple related claims if they come from the same source.
6. For data listed in your tables, the general best way is to put the citation marks at the end of
   the respective data cells. You may use a single citation for a whole table only when _ALL_
   the data in the table comes from the same source, and make sure to state the fact that the same
   citation supports the whole table clearly at your citation site.
7. Your output should be a standalone report, and the user does not have access to any of the
   intermediate artifacts produced by your coworkers under '/workspace', thus citing local sources
   like the intermediate reports by your coworkers under '/workspace' is **STRICTLY PROHIBITED**.
   You must _ONLY_ use references in the '[[cite:id]]' form and nothing else, and consequently,
   you should not make claims that are not supported by these numbered articles.
```

```
### Consistency

Make sure that the report contains no logical inconsistency or factual contradiction.
1. The claims in the report should not contradict themselves. A common mistake is to state a range
   in one part of the report yet calculating with only the upper/lower limit of the range in other
   claims. If such approximation is needed, make sure to include phrases like "at least" or
   "at most" to resolve the range/point conflict.
2. Sometimes, different data sources might claim contradicting facts or data. In this case, prefer
   the data from official sources if available, and clearly point out the situation with the
   conflicting data.
```