Rethinking Agentic Search with ReLex: Is Lexical Retrieval Sufficient?

Anonymous ACL submission

Abstract

Does a lexical retriever suffice as large language models (LLMs) become more capable in an agentic loop? This question naturally arises when building deep research systems. We revisit it by pairing BM25 with frontier LLMs that have better reasoning and tool-use abilities. To support researchers asking the same question, we introduce ReLex, a search agent equipped with three tools for retrieving, browsing, and reading documents. Our results show that, on BrowseComp-Plus (BCP), a well-configured lexical retriever with sufficient retrieval depth can support effective deep research when paired with more capable LLMs. Specifically, ReLex with `gpt-5.5` achieves 83.1% answer accuracy and 94.7% surfaced evidence recall, outperforming released search agents that use dense retrievers. Controlled ablations further show that BM25 tuning improves answer accuracy by 18.0% and surfaced evidence recall by 11.1% over the default BM25 setting, while increasing retrieval depth further improves surfaced evidence recall by 25.3% over the shallow-retrieval setting. All our code and datasets will be released publicly upon paper acceptance.

1 Introduction

Retriever effectiveness is widely treated as the hard ceiling of information-seeking systems such as Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) and deep research (White, 2024; Team et al., 2025; Li et al., 2025). This view has motivated retrievers that can reason over complex queries (Zhang et al., 2025; Shao et al., 2025; Chen et al., 2026), rather than relying on vanilla lexical retrievers such as BM25. However, as frontier LLMs become increasingly capable of reasoning and tool use, these systems increasingly operate through an agentic loop, where LLMs receive feedback from their environments (Yao et al., 2023). Under this interaction pattern, retriever effectiveness may no longer be the only component worth optimizing at all costs. We therefore revisit a central question in building such systems: *Does a lexical retriever suffice as LLMs become more capable in an agentic loop?* Without answering it, we again risk overemphasizing retriever improvement while overlooking other opportunities (Yang et al., 2019).

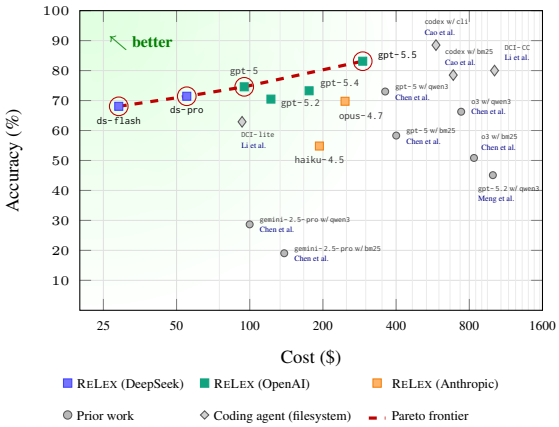

Figure 1: Accuracy vs. Cost trade-off on BrowseComp-Plus. `ds` abbreviates `deepseek-v4` model, and `qwen3` abbreviates `qwen3-embed-8b` retriever. All ReLex systems are paired with BM25.[1]

We first ask whether previous lexical baselines were ill-configured rather than inherently weak. To isolate retrieval capacity from agent behavior, we tune BM25 parameters and retrieval depth, reducing the chance that relevant documents are discarded before LLM inspection. As Figure 1 previews, this simple lexical foundation, when paired with a capable agentic loop, already yields a favorable accuracy–cost trade-off.

Making this pairing effective requires more than returning a deeper ranked list: because LLMs are often constrained by limited context windows, they need an interface for deciding what to inspect and retain. We therefore intro-

[1] Results from Cao et al. (2026) are based on sampled evaluations and provided for reference. See Section 7.

duce RELEX,[2] a deliberately minimal search agent equipped with three tools—`search`, `read_search_results`, and `read_document`—that separate retrieval, result browsing, and document reading. Through this interface, RELEX caches retrieved rankings, explores deeper results, and selectively brings evidence into the LLM context.

In addition, deep research is inherently time-sensitive, requiring systems to balance answer quality against realistic latency and cost constraints. To better match user-facing deep research workloads, we use time-budget steering instead of the fixed iteration cap used in prior work (Chen et al., 2025; Meng et al., 2026), allowing the agent to terminate under a hard wall-clock time budget.

On BrowseComp-Plus (BCP) (Chen et al., 2025), under time-budget steering, RELEX with `gpt-5.5` achieves 83.1% answer accuracy and 94.7% surfaced evidence recall, outperforming released search agents based on dense retrievers. Ablations show that BM25 tuning improves answer accuracy by 18.0% and surfaced evidence recall by 11.1% over the default BM25 setting, while increasing retrieval depth further improves surfaced evidence recall by 25.3% over the shallow-retrieval setting.

Finally, RELEX benefits from a prefix-cache-friendly agentic loop, allowing many repeated input tokens to be served from cache and making prefix caching central to its cost efficiency. BM25 tuning and wall-clock time budgeting provide additional cost savings. Figure 1 summarizes the resulting accuracy–cost trade-off, showing that RELEX consistently lies in the efficient region.

Overall, the experiments on BCP suggest that, with more capable LLMs in an agentic loop, BM25 can be sufficient for effective deep research when it is well-configured, used at sufficient retrieval depth, and paired with a tool interface that helps the agent manage cached retrieval ranking. Our contributions are as follows:

- We introduce RELEX, a search agent that manages cached rankings for deeper retrieval and selective evidence acquisition.
- We reassess BM25 on BCP by separating lexical retrieval capacity from configuration and retrieval depth, showing that both greatly affect answer accuracy and evidence recall.
- We show that, on BCP, BM25-based search agents can match or outperform dense-retriever agents at 3.3×–10× lower evaluation cost.

[2]Anonymized for review purpose.

2 Related Work

Deep Research. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) grounds answer generation in external documents, but complex information-seeking tasks often require more than a single retrieval step. They involve iterative search, document inspection, and evidence synthesis across multiple sources, motivating search agents that extend RAG into multi-step retrieval-and-reasoning processes (Shi et al., 2025; Team et al., 2025; Asai et al., 2024). Recent work has improved these agents through reasoning enhancement (Zhang et al., 2026; Wen et al., 2026; Shao et al., 2024) and tool-use optimization (Wu et al., 2024; Qian et al., 2026). As these systems become more capable, their evaluation has also shifted toward more realistic research settings: Benchmarks such as BrowseComp (Wei et al., 2025) and BrowseComp-ZH (Zhou et al., 2025) evaluate open-web browsing ability, while BCP provides a fixed-corpus setting for controlled comparison of search agents (Chen et al., 2025). These benchmarks make retrieval and evidence interaction central to agent performance, raising the question of how much observed performance depends on the retrieval model itself.

Retriever for Deep Research. Search agents often use dense retrievers, sparse retrievers, and rerankers to improve early-stage ranking accuracy (Zhang et al., 2025). Recent reasoning-aware retrievers further target complex queries by capturing implicit intent and resolving multi-hop information needs (Shao et al., 2025; Chen et al., 2026). By contrast, lexical methods such as BM25 are commonly used as non-dense baselines, and their weaker end-to-end performance is often attributed to semantic mismatch.

Related analyses have examined retrieval components in search agents: SAGE (Hu et al., 2026) studies retrieval effectiveness with LLM-based retrievers, while other work (Sharifymoghaddam and Lin, 2026; Meng et al., 2026) studies how reranking and document ordering affect search agents in deep research. However, these studies do not isolate whether BM25's apparent weakness comes from lexical retrieval itself or surrounding design choices such as configuration and retrieval depth. Our work addresses this gap by separating BM25 capacity from these surrounding choices. We therefore revisit the question: *Does a lexical retriever suffice as LLMs become more capable in an agentic loop?*

3 RELEX

RELEX is a search agent for isolating agent–retriever interaction. The LLM agent runs a ReAct loop, while a retrieval controller mediates all access to an ANSERINI BM25 backend. This controller is the main isolation point: it exposes a constrained tool API, maintains cached ranking from retrieval, and controls how retrieved evidence enters the agent context. Appendix A gives the exact prompt, backend configuration, cache behavior, and termination policy used in our experiments.

3.1 Preliminary

We formulate deep research as an iterative retrieval-and-reasoning problem in a ReAct-style agentic loop (Yao et al., 2023). Given a user query, a search agent, commonly materialized by an LLM, iteratively reasons, calls tools, observes their outputs, and eventually returns a final response.

The agent operates over an interaction trajectory:

$$\mathcal{H}_T = (\tau_1, a_1, o_1, \ldots, \tau_t, a_t, o_t, \ldots, \tau_T, a_T)$$

where τ_t is the reasoning trace generated at turn t, a_t is the action selected by the agent, and o_t is the observation returned by the environment. At each turn, the agent policy π samples a reasoning trace and an action conditioned on the previous interaction history:

$$\tau_t, a_t \sim \pi(\cdot \mid \mathcal{H}_{t-1})$$

For intermediate turns $(t < T)$, the action a_t is a tool call or reasoning, and the environment returns an observation o_t. The final action a_T produces the response and terminates the loop. We treat the outputs as both the final response R and the set of documents D observed during the retrieval process.

3.2 Components

RELEX has three design elements. First, it separates retrieval, result browsing, and document reading into distinct tool calls. Second, it records the agent's interaction trajectory at multiple evidence-access levels. Third, it uses time-budget steering to ensure that the agent answers before timeout.

Retrieval Controller. Naive search agents often use shallow retrieval settings that insert the full texts of all ranked documents directly into the context window. This removes the need for explicit evidence selection, but conflates retrieval depth with context management and makes deeper retrieval impractical. In contrast, RELEX leverages LLM tool use through separate tools for retrieval, browsing, and reading, allowing the agent to cache ranked documents and selectively control which evidence enters its context window. Our tool interface follows the workflow in Appendix A.2: search first, browse the cached ranking when useful, and selectively read promising documents. The action a can be one of three search-related tool calls:

- `search`: issues a lexical query, retrieves up to 1000 ranked documents, caches the ranking under a session-local `search_id`, and exposes only the top 5 excerpts on the initial call;
- `read_search_results`: browses a cached ranking by `search_id` using rank pagination, allowing the agent to inspect results without issuing a new backend query;
- `read_document`: fetches one document by identifier using line-based pagination, allowing the agent to read only the relevant text pieces.

Trajectory Logging. Each run produces a structured per-query artifact that records the full tool call and reasoning trajectory. Rather than tracking only a single document set returned by the search tool, RELEX records four document sets:

- D_{surfaced}: documents returned by `search`.
- $D_{\text{previewed}}$: documents whose excerpts were shown through `read_search_results`.
- D_{opened}: documents read via `read_document`.
- D_{cited}: documents cited in the final answer.

These tiers distinguish what the system made available, what the agent chose to inspect, and what it ultimately used.

Time-Budget Steering. We impose a timeout rather than a fixed iteration cap on the agentic loop. Each query run is subject to a timeout of T seconds.

To encourage completion before hard termination, RELEX uses a two-stage time-budget policy. At $0.7T$, the system injects a submission steer that instructs the agent to stop using tools and produce its best answer from the evidence collected so far. From that point onward, all three tools are blocked. If the agent still does not submit by time T, the process is terminated and the query is marked as timed out.

4 Experimental Setup

We study our research question—*Does a lexical retriever suffice as LLMs become more capable in*

an agentic loop?—by evaluating search agents that pair different LLMs with either dense retrievers or BM25, measuring both retrieval effectiveness and final answer quality.

4.1 Evaluation Settings

Dataset. We evaluate RELEX on the widely used BCP (Chen et al., 2025), a deep research benchmark consisting of 830 queries and 100,195 documents. On average, each query is associated with 6.1 evidence documents and 2.9 gold documents. Each document contains an average of 5,179.2 words and 32,296.2 characters.

Evidence documents are documents required to answer the query, while gold documents are a stricter subset that both support answering and semantically contain the final answer.

Baselines. We compare RELEX with several representative BCP baseline systems. The released systems pair `o3` and `gpt-5` with either `qwen3-embed-8b` (Zhang et al., 2025) or BM25. We also report the baseline from the work of Meng et al. (2026), which also uses `qwen3-embed-8b` retriever but replaces the reasoning model with `gpt-5.2`. Finally, we report the numbers of AgentIR (Chen et al., 2026), a reasoning-intensive retriever trained for deep research.

Unlike RELEX, which exposes separate `search`, `read_search_results`, and `read_document` tools, the baseline agents use a single retriever tool that directly returns the top-k search results, with $k = 5$. Under our logging protocol, we therefore only consider D_{surfaced} for baseline agents, since they do not use additional tools beyond `search` unless specified otherwise.

Metrics. Following BCP, we report both answer-quality metrics and retrieval-behavior metrics. For answer evaluation, we use an LLM judge. For each query, the judge receives the question, the generated response, and the labeled answer, then determines whether the extracted final answer is semantically equivalent to the correct answer. We use `gpt-5.3-codex`; although this differs from BCP's judge model, it reproduces the BCP evaluation outcome in Section 6. Appendix A.5 gives the exact configuration and prompt. Specifically, we report:

- Accuracy: the fraction of queries whose final answers are judged correct by the LLM judge;
- Calibration Error: the discrepancy between the model's confidence and empirical correctness.

For retrieval evaluation, we report recall over the logged document sets introduced in Section 3.2:

- Surfaced Recall: recall computed over the document set D_{surfaced};
- Previewed Recall: recall computed over the document set $D_{\text{previewed}}$;
- Behavior Recall: recall computed over the union of the document sets $D_{\text{opened}} \cup D_{\text{cited}}$.

4.2 RELEX Implementation Details

Retriever. Documents are indexed using BM25 via ANSERINI (Yang et al., 2018) over the BCP corpus. We use tuned BM25 parameters with $k_1 = 25$ and $b = 1$, which are selected to better support long-document retrieval in a deep research setting. The BM25 retrieval depth is set to 1000. This retrieval configuration is held fixed across all RELEX runs. Further details on BM25 tuning are provided in Section 6.

Models. We evaluate RELEX using a diverse set of frontier LLMs with better reasoning and tool-use abilities, reflecting the class of sufficiently capable LLMs considered in our research question. For DeepSeek, we use `deepseek-v4-flash` and `deepseek-v4-pro`. For Anthropic, we use `claude-haiku-4.5` and `claude-opus-4.7`. For OpenAI, we use `gpt-5`, `gpt-5.2`, `gpt-5.4`, `gpt-5.4-mini`, and `gpt-5.5`. This model set across different providers allows us to evaluate whether RELEX's design remains effective across different LLM families and scales.

Agent Harness. We build RELEX on top of PI (Zechner and contributors, 2025), a minimal but malleable agent harness without sub-agents or additional orchestration modules. We repurpose PI as our search agent by removing its system prompts and built-in tools, then design our own. We configure it with our deep research prompt that instructs it to issue queries, inspect retrieval results through our tools, and synthesize an answer. Per-query timeout is set to $T = 300$ seconds.

5 Main Experimental Results

Answer Quality. Table 1 reports the answer accuracy of all systems. We first compare systems that use the same LLM, `gpt-5`, with BM25. Under this setting, RELEX improves answer accuracy from 58.3% for the released `gpt-5 + BM25` baseline in row (c) to 74.6% in row (i). It also slightly exceeds the released `gpt-5 + qwen3-embed-8b`

	LLM	Retriever	Answer Quality			Surfaced Recall		Previewed Recall		Behavior Recall	
			Acc.	Calib.	Cost ($)	Evi.	Gold	Evi.	Gold	Evi.	Gold
Chen et al. (2025)†											
(a)	o3	bm25	50.8	39.1	836.4	56.6	61.7	-	-	-	-
(b)	o3	qwen3-embed-8b	66.3	32.7	740.8	73.2	76.3	-	-	-	-
(c)	gpt-5	bm25	58.3	13.5	400.4	61.7	66.5	-	-	-	-
(d)	gpt-5	qwen3-embed-8b	73.0	9.7	360.7	79.0	**81.3**	-	-	-	-
Meng et al. (2026)‡											
(e)	gpt-5.2	qwen3-embed-8b	45.1	-	$1 \sim 2$k	-	74.7	-	-	-	-
Chen et al. (2026)‡											
(f)	Tongyi-DR	AgentIR-4B	68.1	-	-	**79.2**	-	-	-	-	-
ReLex (ours)											
(g)	claude-haiku-4.5	bm25	54.8	17.3	193.5	94.1	**95.4**	58.2	60.1	40.9	46.3
(h)	claude-opus-4.7		69.8	10.2	246.6	81.2	86.8	43.3	52.5	30.4	43.0
(i)	gpt-5		74.6	7.2	94.9	90.5	93.8	62.7	70.0	45.7	56.2
(j)	gpt-5.2		70.5	6.2	122.2	89.9	92.8	60.5	67.3	44.8	54.5
(k)	gpt-5.4-mini		68.1	13.7	86.2	91.9	94.1	60.1	65.4	43.1	52.1
(l)	gpt-5.4		73.3	9.2	175.5	93.8	95.3	70.3	66.9	51.8	58.1
(m)	gpt-5.5		**83.1**	15.7	291.6	**94.7**	94.4	73.6	72.9	58.9	56.1
(n)	deepseek-v4-flash		68.1	15.5	**28.9**	94.5	95.7	67.9	69.9	55.2	60.6
(o)	deepseek-v4-pro		71.4	7.0	55.1	91.3	92.5	60.0	63.0	45.4	50.6

Table 1: Answer quality, retrieval behavior, and cost on BCP.

† Results from Chen et al. (2025) are obtained by evaluating their released runs using our own judge.
‡ Results from Meng et al. (2026) and Chen et al. (2026) are taken from the reported numbers in the original papers.

dense-retriever baseline in row (d). This comparison suggests that previously released baselines understate the potential of BM25, and that improving its configuration and increasing retrieval depth makes it a competitive alternative to dense retrievers for deep research.

Among BM25-based systems, ReLex with gpt-5.5 in row (m) achieves the highest accuracy, at 83.13%. This is also the highest accuracy in Table 1, further showing that a BM25-based search agent can achieve highly competitive performance when BM25 is well-configured, used with sufficient retrieval depth, and paired with more capable LLMs in an agentic loop.

On the other hand, using a dense retriever does not necessarily translate into better answer quality. The system from Meng et al. (2026), which pairs gpt-5.2 with qwen3-embed-8b in row (e), achieves only 45.1% accuracy. In contrast, ReLex with the same LLM, gpt-5.2, reaches 70.5% accuracy using BM25 in row (j). This gap suggests that final answer quality is not determined by retriever choice alone, but also by how the agent searches, manages context, and interacts with retrieved documents. The same pattern is also observed for the AgentIR-4B baseline in row (f).

Overall, these results address our research question in terms of final answer quality, showing that BM25 can achieve competitive answer accuracy in deep research when its parameters are well-configured, its retrieval depth is sufficient, and it is paired with more capable LLMs in an agentic loop.

We make two additional observations from the answer-quality results. First, the DeepSeek-based agents provide a strong cost–performance trade-off: deepseek-v4-flash achieves 68.1% accuracy with only $28.9 total cost in row (n). Second, there is a substantial gap between claude-opus-4.7 and gpt-5.5, despite both being high-cost frontier models. claude-opus-4.7 achieves 69.8% accuracy in row (h), while gpt-5.5 reaches 83.1% in row (m). We further analyze their behavioral differences in Appendix C.

Retrieval Behavior. Table 1 compares retrieval behavior across systems. For surfaced recall, all ReLex variants substantially outperform the released baselines, with most runs achieving over 90% recall on both evidence and gold documents. Notably, gpt-5.5 + BM25 in row (m) reaches 94.7 and 94.4 on the evidence and gold documents, respectively, compared to 79.21 and 81.34 for the dense-retriever baselines of rows (f) and (d).

Under the same gpt-5 + BM25 setting, ReLex also exposes more evidence to the agent. Compared with the released baseline in row (c), ReLex in row (i) improves previewed recall of gold doc-

uments from 66.5 to 70.0, while maintaining similar previewed evidence-document recall. With gpt-5.5, (m) further achieves the highest previewed recall among BM25-based systems, reaching 73.6 for evidence documents and 72.9 for gold documents. Because agents can often synthesize answers from previewed excerpts without calling read_document, higher previewed recall need not yield proportional gains in behavior recall.

Overall, these results answer our research question from the perspective of retrieval effectiveness: a lexical retriever can retrieve the evidence needed for deep research when paired with more capable LLMs in an agentic loop, provided that BM25 is well-configured and used with sufficient retrieval depth. At the same time, the gap between surfaced and previewed recall shows that the remaining challenge lies in helping the agent navigate cached ranking and allocate context to the right evidence.

Cost Analysis. Prior work on BCP has shown that deep research evaluation can be expensive (Chen et al., 2025; Meng et al., 2026). Meng et al. (2026), for example, report that one gpt-5.2 experiment in row (e) costs roughly $1000–$2000 USD because many queries require many iterations for an agent to reach its conclusion.

RELEX substantially lowers this cost while preserving competitive performance. Under the same gpt-5 + BM25 setting, RELEX reduces the cost from $400.4 in the released baseline row (c) to $94.9 in row (i). Even when paired with more expensive LLMs, RELEX remains cost-efficient. For example, gpt-5.5 + BM25 in row (m) achieves the highest accuracy in the table at a cost of $291.6, which is still lower than the released gpt-5 baselines in rows (c) and (d). Similarly, RELEX with gpt-5.2 costs $122.2, far below the roughly $1000–$2000 reported for the gpt-5.2 experiment in row (e).

These cost reductions matter because deep research studies depend on repeated evaluation. By lowering the cost of each full benchmark run, RELEX makes ablations and future studies of deep research more practical. We further analyze the factors contributing to the cost savings in Section 6.

Tool Usage. Table 2 further compares how different search agents allocate their tool budget. The results show that, except for the claude-haiku-4.5-based agent, most RELEX systems use a similar or smaller number of total tool calls compared to the BCP released baselines. This pattern is especially clear under the same gpt-5 + BM25 setting. Compared with the released baseline, which uses 23.2 tool calls per query, RELEX uses only 15.2. In contrast, the gpt-5.2 + qwen3-embed-8b system reported by Meng et al. (2026) uses a dense retriever but requires substantially more tool calls, 73.8 per query, while not achieving better answer accuracy than RELEX. This indicates that replacing dense retrievers with BM25 does not necessarily increase the number of tool calls, while still maintaining competitive retrieval effectiveness and answer accuracy, as shown in Table 1.

System	Total	Search	Read	Browse
Chen et al. (2025)				
o3 + BM25	25.9	25.9	–	–
o3 + qwen3-embed-8b	24.0	24.0	–	–
gpt-5 + BM25	23.2	23.2	–	–
gpt-5 + qwen3-embed-8b	21.7	21.7	–	–
Chen et al. (2026)				
Tongyi-DR + AgentIR-4B	27.9	24.5	3.4	–
Meng et al. (2026)				
gpt-5.2 + qwen3-embed-8b	73.8	73.8	–	–
RELEX (ours)				
claude-haiku-4.5 + BM25	41.3	31.8	8.8	0.6
claude-opus-4.7 + BM25	9.0	6.8	2.1	0.1
gpt-5 + BM25	15.2	11.1	3.9	0.2
gpt-5.2 + BM25	17.2	12.6	3.9	0.7
gpt-5.4-mini + BM25	24.7	20.2	4.2	0.3
gpt-5.4 + BM25	23.1	17.9	4.3	0.9
gpt-5.5 + BM25	19.3	13.5	5.0	0.8
deepseek-v4-flash + BM25	33.2	26.1	7.0	0.1
deepseek-v4-pro + BM25	18.6	14.2	4.3	0.1

Table 2: Average tool calls per query for all systems

6 Ablation Study

Retrieval Depth. We vary the number of documents k returned by search to measure the effect of retrieval depth on recall.

Figure 2 shows that increasing k from 5 to 100 substantially improves surfaced recall from 70.5% to 86.22%. Previewed recall also improves over the same range, from 70.5% to 74.1%. When k is further increased to 1000, surfaced recall continues to rise, reaching 95.8%. In contrast, previewed recall does not continue to improve with larger k and reaches its highest value of 74.7% at $k = 50$.

These results show that retrieval depth is a major driver of surfaced recall in RELEX. Increasing k gives BM25 enough first-stage capacity to place relevant evidence documents in the cached ranking, but the saturation of previewed recall suggests that larger cached rankings do not automatically lead the agent to inspect more relevant evidence.

Tuning BM25 for Long Documents. BM25 defaults are usually chosen for shorter documents,

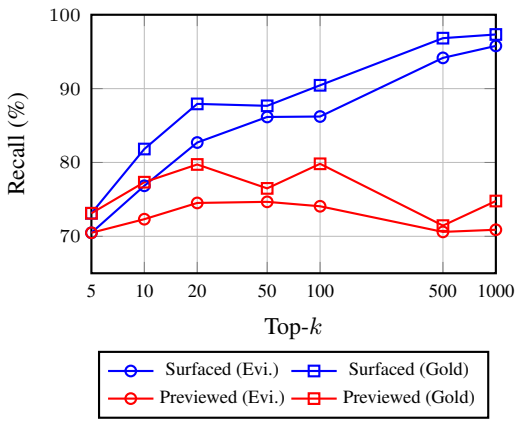

Figure 2: Effect of the number of documents (k) returned by the `search` tool in RELEX with `gpt-5.4`

but documents in BCP are long and noisy. We therefore test whether RELEX's gains depend on retuning BM25 for this setting.

We sample 100 queries from BCP and perform a grid search over k_1 and b. Figure 3 shows that ANSERINI's default setting ($k_1 = 0.9, b = 0.4$) is placed in a low-performing region. Stronger configurations use larger k_1 and higher b, with the best setting near $k_1 = 16$ and $b = 1.0$. This pattern suggests that the default BM25 setting is underfit for long-document evidence search.

The first and second columns in Table 3 show the downstream effect on a 100-query subset of BCP. The default setting uses $k_1 = 0.9$ and $b = 0.4$, while the tuned setting uses $k_1 = 25$ and $b = 1$, following Table 1. We also tested the best grid-search setting, $k_1 = 16$ and $b = 1$, and observed similar results. Tuning raises accuracy from 64.0% to 82.0% (81.1% with the best grid-search setting) and surfaced recall from 84.6% to 95.7% (94.0% with the best grid-search setting). Previewed recall and behavior recall also improve, showing that better first-stage ranking gives the agent more useful evidence to inspect.

This sensitivity is unsurprising given the length of documents in BCP: the median document has ~2k tokens, and the 90th percentile has ~14k tokens (cf. Figure 4 in Chen et al. (2025)). At this length, relevant evidence is often buried in irrelevant text, making length normalization and term-frequency saturation matter more than in passage retrieval. ANSERINI's default BM25 setting is tuned for that shorter-document regime, whereas tuned parameters better match long-document evidence search. This adaptation improves both evidence recall and final answer accuracy, consistent with the prior findings (Meng et al., 2026).

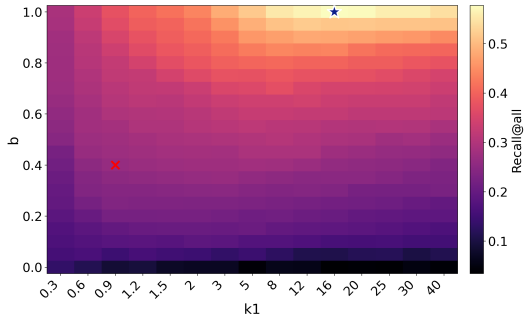

Figure 3: Grid search results of BM25 tuning over different k_1 and b combinations. ★ indicates the best-performing configuration, while × marks ANSERINI's default setting: ($k_1 = 0.9, b = 0.4$).

	Default	Tuned	Tuned	Tuned
BM25 (k_1, b)	(0.9, 0.4)	(25, 1)	(25, 1)	(25, 1)
Term. Condition	Timeout 300s	Timeout 300s	MaxIter100	Timeout 3600s
Answer quality				
Overall (%)	64.0	82.0	76.0	83.0
Completed (%)	66.7	83.7	92.7	83.0
Retrieval effectiveness				
Surfaced (%)	84.6	95.7	97.3	96.8
Previewed (%)	50.7	70.4	73.5	79.0
Behavior (%)	36.4	52.2	56.5	60.0
Behavioral diagnosis				
Avg. Tool Calls	22.6	23.1	27.3	27.1
Med. Total Tokens	281,324.0	271,140.5	263,880.5	218,085.0
Med. Cached Tokens	228,736.0	227,968.0	215,552.0	186,112.0
Cost ($)	24.1	21.7	24.9	26.3

Table 3: Performance, cost, and tool-use comparison between default/tuned BM25 under different termination policies tested by RELEX with `gpt-5.4`.

Cost-Efficiency. To better understand which designs in RELEX improve cost efficiency, we focus on three factors: (1) BM25 tuning, (2) termination policy, and (3) prefix cache.

We first examine the effect of BM25 tuning. The first and second columns of Table 3 show that, under the same termination policy, cost can be slightly reduced with a better retriever. This phenomenon is consistent with the baseline results in Table 1, from row (a) to row (d).

For the termination policy, we compare three settings: (1) a 300-second timeout, which is the standard setting in RELEX; (2) a maximum of 100 iterations, which aligns with the baseline setting (Chen et al., 2025; Meng et al., 2026); and (3) a 3600-second timeout, which is an extremely large timeout used to simulate the absence of a practical termination policy. According to the second through last columns of Table 3, replacing the 100-iteration cap with the 300-second timeout slightly reduces cost, from $24.9 to $21.7, while substantially increasing the timeout to 3600 seconds does not dramatically increase the cost.

Regarding prefix cache, RELEX inherits PI's agentic loop, which structures the interaction his-

	BCP (Chen et al., 2025)	Partial RELEX	Partial RELEX	Full RELEX
Tools available	1	1	1	3
Tuned BM25	–	–	✓	✓
Retrieval Depth	5	5	5	1000
Agent Harness	Custom	PI	PI	PI
BM25 (k_1, b)	$(0.9, 0.4)$	$(0.9, 0.4)$	$(25, 1)$	$(25, 1)$
Answer quality				
Overall (%)	55.9	54.3	70.7	74.6
Retrieval effectiveness				
Surfaced (%)	61.7	49.9	69.1	90.5
Previewed (%)	–	–	–	62.7
Behavior (%)	–	–	–	45.7
Cost ($)	400.4	140.6	140.4	94.9

Table 4: Ablation of RELEX components on BCP. All RELEX variants use `gpt-5` to match BCP settings. BCP results are from Chen et al. (2025).

tory and enables prefix-cache-friendly configuration by default. Although Table 3 does not directly isolate the cost reduction from prefix caching, all settings exhibit high cached-token ratios: around 82% to 90% of total tokens are served from cache. Given the lower price of cached tokens, this suggests that the prefix-cache-friendly loop is a major contributor to RELEX's overall cost efficiency.

Reproducing BCP. We reproduce the released BCP search-only setup with RELEX, keeping the system prompt, search-only tool interface, top-5 retrieval depth, and default BM25 parameters fixed. As shown in the first two columns of Table 4, RELEX achieves comparable answer accuracy to BCP, 54.3% versus 55.9%, while reducing cost from $400.4 to $140.6. Surfaced recall is lower, 49.9% versus 61.7%, likely due in part to differences in judge models, but the similar answer accuracy suggests that RELEX provides a faithful and cheaper reproduction of the BCP setup.

Tuning BM25 gives the largest improvement, increasing answer accuracy from 54.3% to 70.7% and surfaced recall from 49.9% to 69.1%, with essentially unchanged cost. Expanding from the tuned top-5 search-only setting to top-1000 setting with three tools further improves accuracy from 70.7% to 74.6% and surfaced recall from 69.1% to 90.5%, while reducing cost from $140.4 to $94.9. These results suggest that RELEX improves cost efficiency by letting the agent explore cached ranking, though further gains may require stronger support for selecting evidence from that ranking.

7 Discussion

To Retrieve or Not To Retrieve. Recent work by Cao et al. (2026) and Li et al. (2026) shows that coding agents can perform well on BCP by turning deep research into file-system navigation: the agent searches a locally materialized document set with command-line tools (cf. Figure 1: `codex w/cli`, `DCI-lite`, and `DCI-CC`). This setup is complementary to retrieval-based search agents, but it assumes that the relevant search space can be localized in advance, which is difficult for large, dynamic corpora. Moreover, some reported results use subsampled evaluations to reduce cost (Cao et al., 2026). For these reasons, we report these systems only as reference points in Figure 1, rather than as direct comparisons in Table 1.

Nevertheless, their systems can be viewed as testing a setting where relevant documents are already available locally and the agent only needs to identify the needed context through simple interaction mechanisms. Our results show a complementary point that, with proper configuration and sufficient retrieval depth, BM25 can already provide a high-recall candidate set for the agent to inspect.

8 Conclusion

We asked: "Does a lexical retriever suffice as LLMs become more capable in an agentic loop?" On BCP, our results suggest yes: a well-configured lexical retriever with sufficient depth can support effective deep research when paired with more capable LLM agents. In our experiments, it matches or exceeds released dense-retriever baselines, while our prefix-cache-friendly agent harness substantially reduces full-benchmark cost. The broader lesson is not that BM25 is enough everywhere, but that under-configured baselines can hide weak agent–retriever interaction and miss the opportunities in designing more effective and cost-efficient search agents.

For us as a community interested in continuing to push the boundaries of deep research and information-seeking systems, the most useful signal is where RELEX fails. It often surfaces the right evidence documents, but the agent, with its pre-trained policies and behaviors in this era, does not always browse, open, or use them effectively. That failure mode shifts the question of system improvements from *Can the retriever find the evidence?* to *Can the agent recognize and spend context on the evidence it has already been given?* Future progress may therefore come less from pulling the retriever lever harder, and more from designing tools that help agents navigate evidence documents with better judgments.

9 Limitations

RELEX shows that a well-configured BM25 retriever, combined with a carefully designed tool interface, can surface high-recall evidence. However, our study has several limitations. First, RELEX still relies on a relatively minimal interface for navigating the cached ranking, resulting in substantially lower previewed and behavior recall than surfaced recall. Second, our evaluation is limited to BCP, leaving open the question of whether the current agent harness generalizes to other settings, including multilingual queries and domain-specific scenarios. Finally, although using a timeout as the termination policy provides a simple way to control execution, queries with different levels of complexity may require different amounts of research time. A fixed time budget may therefore fail to balance effectiveness, latency, and monetary cost, leading either to insufficient exploration for difficult queries or unnecessary computation for simpler ones.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the Twelfth International Conference on Learning Representations*.

Weili Cao, Xunjian Yin, Bhuwan Dhingra, and Shuyan Zhou. 2026. Coding agents are effective long-context processors. *Preprint arXiv:2603.20432*.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Jimmy Lin, Akari Asai, and Victor Zhong. 2026. AgentIR: Reasoning-aware retrieval for deep research agents. *Preprint, arXiv:2603.04384*.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. 2025. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *Preprint, arXiv:2508.06600*.

Tiansheng Hu, Yilun Zhao, Canyu Zhang, Arman Cohan, and Chen Zhao. 2026. SAGE: Benchmarking and improving retrieval for deep research agents. *Preprint arXiv:2602.05975*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438.

Zhuofeng Li, Haoxiang Zhang, Cong Wei, Pan Lu, Ping Nie, Yi Lu, Yuyang Bai, Shangbin Feng, Hangxiao Zhu, Ming Zhong, Yuyu Zhang, Jianwen Xie, Yejin Choi, James Zou, Jiawei Han, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. 2026. Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction. *Preprint, arXiv:2605.05242*.

Chuan Meng, Litu Ou, Sean MacAvaney, and Jeff Dalton. 2026. Revisiting text ranking in deep research. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2026. ToolRL: Reward is all tool learning needs. In *Proceedings of 39th Annual Conference on Neural Information Processing Systems*.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. 2025. ReasonIR: Training retrievers for reasoning tasks. *Preprint, arXiv:2504.20595*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *Preprint arXiv:2402.03300*.

Sahel Sharifymoghaddam and Jimmy Lin. 2026. Rerank before you reason: Analyzing reranking tradeoffs through effective token cost in deep search agents. *Preprint, arXiv:2601.14224*.

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, and 7 others. 2025. Deep research: A systematic survey. *Preprint arXiv:2512.02038*.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, and 38 others. 2025. Tongyi DeepResearch technical report. *Preprint, arXiv:2510.24701*.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. BrowseComp: A simple yet challenging benchmark for browsing agents. *Preprint, arXiv:2504.12516*.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2026. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *Proceedings of the 14th International Conference on Learning Representations*.

Ryen W. White. 2024. Advancing the search frontier with AI agents. *Commun. ACM*, 67(9):54–65.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N. Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. 2024. AvaTaR: Optimizing LLM agents for tool usage via contrastive reasoning. In *Proceedings of the Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using Lucene. *J. Data and Information Quality*, 10(4).

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically examining the" neural hype" weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 1129–1132.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*.

Mario Zechner and contributors. 2025. PI-MONO: AI agent toolkit (coding agent CLI, unified LLM API, TUI & web UI, Slack bot, vLLM pods). `https://github.com/badlogic/pi-mono`. GitHub repository.

Wenlin Zhang, Kuicai Dong, Junyi Li, Yingyi Zhang, Xiaopeng Li, Pengyue Jia, Yi Wen, Derong Xu, Maolin Wang, Yichao Wang, Yong Liu, and Xiangyu Zhao. 2026. To search or not to search: Aligning the decision boundary of deep search agents via causal intervention. In *Proceedings of the ACM Web Conference 2026*, page 2049–2059.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint, arXiv:2506.05176*.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. 2025. BrowseComp-ZH: Benchmarking web browsing ability of large language models in Chinese. *Preprint, arXiv:2504.19314*.

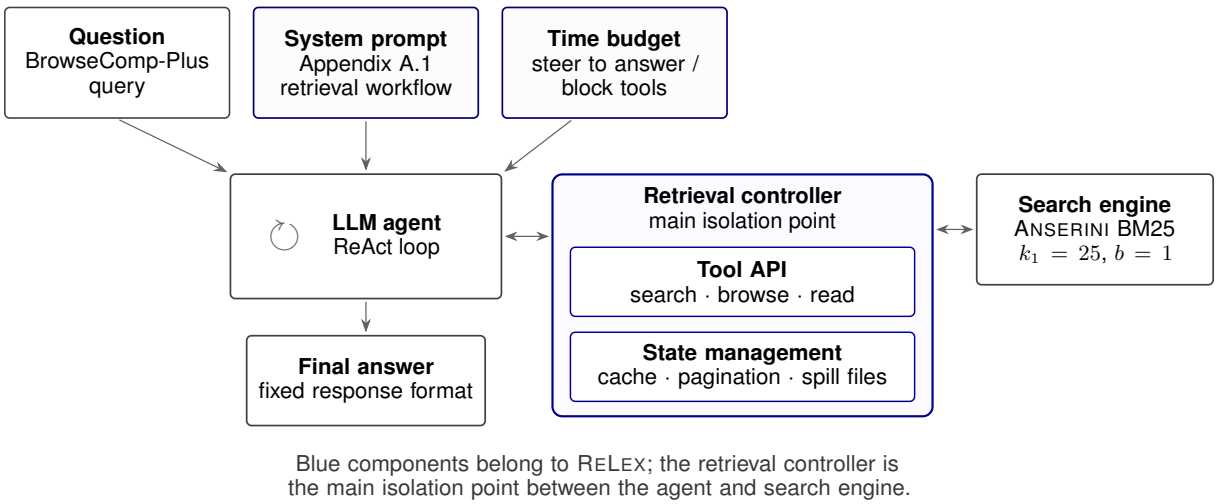

Figure 4: System architecture of RELEX. The system includes the retrieval workflow prompt, time-budget steering policy, and retrieval controller. The LLM agent interacts with ANSERINI only through the controller, which exposes a constrained tool API and maintains local retrieval state for controlled, paginated BM25 retrieval.

A System Setup

This appendix documents the exact RELEX setup used in our experiments. We organize the setup by prompt, tool interface, backend configuration, and termination policy. Placeholders in curly braces, such as `{Question}`, are replaced at runtime.

A.1 Agent Prompt

Search Agent Prompt

```
You are a deep research agent answering a question using
only the provided tools.

Workflow:
1. Use search with a concise raw query string based on the
original question.
2. Prefer short lexical searches over long natural-language
rewrites.
3. Browse the current ranking with read_search_results
before repeatedly rewriting the query.
4. If a promising candidate document appears in the
ranking, inspect it with read_document.
5. When reading a document, start with offset=1 and a
moderate limit. If it is truncated and still relevant,
continue reading the same document.
6. Use search refinements only when they add a genuinely
new clue from what you already saw.
7. Every call to search, read_search_results, and
read_document must include reason as the first argument.
Keep it specific, under 100 words, and focused on the clue,
gap, candidate, or ranking issue.
8. As soon as you have enough evidence, stop using tools
and answer in plain assistant text.
9. Your final response must use exactly this format:
   Explanation: {your explanation for your final answer.
   Cite supporting docids inline in square brackets [] at
   the end of sentences when possible, for example [123].}
   Exact Answer: {your succinct, final answer}
   Confidence: {your confidence score between 0% and 100%}
10. If you later receive a user steer telling you to submit
now, stop using tools immediately and answer right away
with the exact final response format below. Do not do more
research after that steer.
11. Keep Exact Answer concise and directly responsive to
the question.

Question: {Question}
```

A.2 Retrieval Tool Interface

search. The `search` tool sends a raw lexical query to the configured retrieval backend. It takes the following arguments:

- `reason`: brief rationale, supplied first, at most 100 words.
- `query`: raw query string; not a structured object and not raw Lucene syntax.

In our BM25 runs, `query_mode` is set to `plain`, the backend request limit is 1000 hits, and the initial displayed page contains ranks 1–5. The output includes `search_id`, the formatted first page, cached document IDs, displayed document IDs, and timing/truncation metadata.

read_search_results. The `read_search_results` tool browses an existing cached ranking by `search_id` without issuing a new backend query. It takes the following arguments:

- `reason`: brief rationale, supplied first, at most 100 words.
- `search_id`: identifier returned by `search`.
- `offset`: optional 1-indexed rank offset; default is 6.
- `limit`: optional num. of ranked hits to show; default is 10.

The output includes the formatted page from the cached ranking, displayed document IDs, `next_offset` when more hits remain, and truncation metadata if the formatted page is too large.

read_document. The `read_document` tool reads one backend document by `docid` in line-based chunks. It takes the following arguments:

- `reason`: brief rationale, supplied first, at most 100 words.
- `docid`: document id from `search`/`read_search_results`.
- `offset`: optional 1-indexed line offset; default is 1.
- `limit`: optional maximum number of lines; default is 200.

For a found document, the output includes formatted document lines, the returned line range, total line count, backend truncation flag, `next_offset` when available, timing metadata, and output-truncation metadata when applicable.

A.3 Backend and Cache Configuration

Retrieval Backend Configuration. In this paper, we use ANSERINI BM25 as the retrieval backend over the BrowseComp-Plus corpus, with BM25 parameters $k_1 = 25$ and $b = 1$. The pi-search configuration requires `PI_SEARCH_EXTENSION_CONFIG`, which must be supplied by the caller. Our experiments use the ANSERINI BM25 backend kind. The pi-search configuration schema supports the following backend kinds:

- ANSERINI-BM25 over TCP:

```
{
  "backend": {
    "kind": "anserini-bm25",
    "transport": {
      "kind": "tcp",
      "host": host,
      "port": port
    }
  }
}
```

- ANSERINI-BM25 over stdio:

```
{
  "backend": {
    "kind": "anserini-bm25",
    "transport": {
      "kind": "stdio",
      "indexPath": indexPath
    }
  }
}
```

- http-json:

```
{
  "backend": {
    "kind": "http-json",
    "capabilities": capabilities
    "endpoints": endpoints
  }
}
```

- Mock backend:

```
{
  "backend": {
    "kind": "mock",
    "documents": documents
  }
}
```

Search Cache, Pagination, and Spill Files. Each `search` call receives a session-local `search_id`, such as `s1`, `s2`, and so on. Each `search_id` stores the raw query, query mode, and retrieved ranking. At most 32 `search_id` entries are retained, with older entries evicted first.

For retrieval depth and pagination, `search` caches up to 1000 backend hits but initially displays only ranks 1–5.

For spill files, tool output is first formatted for the agent context. If the formatted output exceeds Pi's line or byte truncation limits, the visible output is truncated and the complete search page or document chunk is saved under a temporary pi-search spill directory. The temporary spill directory is cleaned up on session shutdown or process exit.

A.4 Time-Budget Termination Policy

In the main experiments, the per-query timeout is $T = 300$ seconds. The timer is enabled only when `TIMEOUT_SECONDS` is set to a positive finite value. The submit-now trigger fires after `floor(0.7*TIMEOUT_SECONDS*1000)` ms.

When the timer fires while the agent is active, the system applies the following steering policy:

- It sends a user steer indicating that the time budget is nearly exhausted.
- It instructs the agent to stop using tools and submit its best answer immediately.
- It blocks later calls to `search`, `read_search_results`, and `read_document`.

A.5 Gold-Answer LLM Judge

We evaluate final-answer correctness with a gold-answer LLM judge implemented as a no-tool PI agent. The judge receives the question, the agent's final response, and the benchmark-provided correct answer. The default judge model is `openai-codex/gpt-5.3-codex`, run through PI in JSON mode with low reasoning effort and a timeout of 180 seconds per query. The execution command is:

```
pi -no-tools -no-session -no-skills -mode json
-model openai-codex/gpt-5.3-codex -thinking low
{judge_prompt}
```

Evaluation outputs are written under `evals/pi_judge`. The ground-truth answers are read

from `data/browsecomp-plus/ground-truth/browsecomp_plus_decrypted.jsonl`. For each query, the evaluator writes a JSON judgment to `per-query/{query_id}_eval.json`, raw event logs to `raw-events/{query_id}.jsonl`, and standard-error logs to `stderr/{query_id}.log`. The judge prompt is as follows:

Gold-Answer Judge Prompt

```
You are an evaluation judge.

Your job is to determine whether the response's final answer
is semantically equivalent to the known correct answer.
Do not solve the question yourself.
Do not use outside knowledge.
Focus only on whether the response's final answer matches
the correct answer.
Allow harmless wording differences, equivalent formatting,
and added correct detail.
For numerical answers, allow small formatting differences
and obvious equivalent forms.
If the response does not contain a final answer you can
extract, set extracted_final_answer to null and correct to
false.

Return exactly one JSON object and nothing else.
Do not wrap the JSON in markdown or code fences.
Use this exact schema:
{
  "extracted_final_answer": string | null,
  "correct_answer": string,
  "reasoning": string,
  "correct": boolean,
  "confidence": number
}

Requirements:
- confidence must be a number between 0 and 100
- correct must be true or false
- repeat the provided correct answer exactly in
correct_answer
- reasoning must explain only whether the extracted final
answer matches the correct answer

Question: {question}

Response:
{response}

Correct answer: {correct_answer}
```

B Model Token Pricing Used in Benchmark Runs

Table 5 reports the token pricing used by our benchmark runs. All prices are in **USD per 1M tokens**.

Table 5: Token pricing used during benchmark execution. All prices are in USD per 1M tokens.

Model	Input	Output	Cache Read
`claude-haiku-4.5`	1.00	5.00	0.10
`claude-opus-4.7`	5.00	25.00	0.50
`gpt-5`	1.25	10.00	0.125
`gpt-5.2`	1.75	14.00	0.175
`gpt-5.3-codex`	1.75	14.00	0.175
`gpt-5.4-mini`	0.75	4.50	0.075
`gpt-5.4`	2.50	15.00	0.25
`gpt-5.5`	5.00	30.00	0.50
`deepseek-v4-flash`	0.14	0.28	0.028
`deepseek-v4-pro`	1.74	3.48	0.145

C Premature Branch Commitment

GPT-5.5 and Claude Opus 4.7 are both frontier models with comparable costs in our experiments, yet they show a substantial performance gap on BCP. Our trajectory analysis shows that both agents sometimes expand queries using internal knowledge or weakly related entities from retrieved documents. The key difference is how they react when such probes are weak or incorrect. When `gpt-5.5` outperforms `claude-opus-4.7`, it often keeps candidate-specific probes reversible and returns to the original clues after failed searches. Claude Opus 4.7, in contrast, often continues searching within the same weak hypothesis. We refer to this failure mode as *premature branch commitment*. We use Query 678 from BCP to illustrate this phenomenon. Given the original query below, Figures 5 and 6 show the query trajectories of `gpt-5.5` and `claude-opus-4.7`, respectively.

Query 678

```
Find the school mentioned in a November 2019 interview by
an artist who grew up in a town with 2019 population around
114,000 and current spelling appearing around 1316;
performed two roles in the same industry; won a late-2010s
award; released a creation in 2019; described their style
as "eclectic"; described social media as "dangerous"; and
said Instagram or Twitter were the best way to connect.
```

In Figure 5, `gpt-5.5` introduces unverified candidate terms, including Warrington, Vinegar Strokes, and Arts Educational, but treats these probes as reversible. After unproductive searches, GPT-5.5 returns to the root constraints, such as the town population and spelling-history clue, and follows the evidence-supported Crawley path. This eventually leads to the correct candidate, Dani Sylvia, after which the query is refined using the original interview constraints.

Figure 6 shows a different pattern. After the first search retrieves a title mentioning Renato Paris and CICAS, Claude Opus 4.7 treats this superficial match as a candidate anchor. It then searches vertically within the same branch, adding unverified terms such as Tom Misch, Italy, Forlì, ICMP, BIMM, and Berklee. These queries deepen the same hypothesis rather than testing alternative interpretations of the original clues.

```
Root: Original query

|-- Step 1
|   Reason: Locate the November 2019 interview containing the exact quoted style/social media phrases.
|   Query:  "eclectic" "social media is dangerous" "best way to connect" interview November 2019 artist
|
|-- Step 2
|   Reason: Search exact dangerous quote with November interview lexical clues, dropping broader social phrase.
|   Query:  "social media is dangerous" "Instagram" "Twitter" "interview" "2019" artist
|
|-- Step 3
|   Reason: Find pages with both eclectic self-description and contact via social platforms from music interviews.
|   Query:  "How would you describe your style" "eclectic" "social media" "dangerous"
|
|-- Step 4
|   Reason: Search exact interview question about style with release-year clue likely in independent artist Q&A.
|   Query:  "describe your style as eclectic" "interview" "2019" musician
|
|-- Step 5
|   Reason: Target the requested answer phrase in the interview: artist says social media dangerous and mentions training school.
|   Query:  "social media" "dangerous" "trained at" "eclectic" "interview"
|
|-- Step 6
|   Reason: Exact dangerous phrase alone may reveal the interview despite noisy broad results.
|   Query:  "social media is dangerous"
|
|-- Step 7
|   Reason: Look for variant wording around social media danger plus eclectic in artist interviews.
|   Query:  "social media" "dangerous" "eclectic" "Instagram" "interview" "November"
|
|-- Step 8
|   Reason: Use the town clue Warrington plus exact social-media/interview terms to identify the artist.
|   Query:  "grew up in Warrington" artist "eclectic" "social media" "dangerous"
|   |
|   `-- Step 9
|       Reason: Search Warrington artist and 2019 interview clue without exact dangerous phrase.
|       Query:  "grew up in Warrington" "interview" "eclectic" "2019"
|
`-- Step 10
    Reason: Identify town from population 114,000 and name-spelling clue.
    Query:  "current spelling" "first appeared" "1316" town population 114000
    |
    |-- Step 11
    |   Reason: Use identified town Crawley with the interview quote to find the artist.
    |   Query:  "grew up in Crawley" "social media" "dangerous" "eclectic"
    |
    |-- Step 12
    |   Reason: Search Crawley plus November 2019 interview and eclectic without exact grew-up phrase.
    |   Query:  "Crawley" "eclectic" "social media" "interview" "November" "2019" artist
    |
    |-- Step 13
    |   Reason: Candidate from Crawley clue: Vinegar Strokes; verify interview wording and training school.
    |   Query:  "Vinegar Strokes" "eclectic" "social media" "dangerous" "trained"
    |   |
    |   `-- Step 14
    |       Reason: Find candidate biography documents for Vinegar Strokes and relevant facts.
    |       Query:  "Vinegar Strokes" "grew up in Crawley" "Arts Educational"
    |
    `-- Step 15
        Reason: Candidate Dani Sylvia matches Crawley, 2019 release, and songwriter award.
        Query:  "Dani Sylvia" "eclectic" "social media" "dangerous" "school"
```

Figure 5: Case-study query trajectory for GPT-5.5.

```
Root: Original query

|-- Step 1
|   Reason: Find the November 2019 interview where artist calls style "eclectic" and social media "dangerous".
|   Query:  interview November 2019 style "eclectic" social media "dangerous"
|   |
|   `-- Step 2
|       Reason: Find the actual Renato Paris CICAS interview content.
|       Query:  Renato Paris CICAS interview eclectic dangerous
|       |-- Step 4
|       |   Reason: Find info about Renato Paris artist hometown school training.
|       |   Query:  Renato Paris musician singer producer hometown
|       |-- Step 5
|       |   Reason: CICAS podcast about Renato Paris.
|       |   Query:  CICAS podcast Renato Paris keyboardist
|       |-- Step 6
|       |   Reason: Find Renato Paris biography or interview content elsewhere in corpus.
|       |   Query:  Renato Paris interview eclectic style
|       |-- Step 7
|       |   Reason: Search for Renato Paris bio---he might be a different artist.
|       |   Query:  "Renato Paris" biography born
|       |-- Step 10
|       |   Reason: Renato Paris Tom Misch keyboardist Forli.
|       |   Query:  Renato Paris Tom Misch keyboards Italy
|       |   |
|       |   `-- Step 11
|       |       Reason: Forlì Italy population history spelling.
|       |       Query:  Forlì Wikipedia population history name origin
|       |       |-- Step 12
|       |       |   Reason: Look for Forlì Wikipedia article in the corpus.
|       |       |   Query:  Forlì city Italy Emilia-Romagna
|       |       |-- Step 16
|       |       |   Reason: Try different angle---artist from Forlì grew up there.
|       |       |   Query:  "grew up in Forlì" musician artist
|       |       `-- Step 17
|       |           Reason: Try searching for transcription of the interview content with Italian connection.
|       |           Query:  Renato Paris singer Italian Forlì conservatory
|       |-- Step 13
|       |   Reason: Search general corpus for Renato Paris music school training.
|       |   Query:  Renato Paris trained school music London
|       |-- Step 15
|       |   Reason: CICAS abbreviation context---what is it.
|       |   Query:  CICAS podcast meaning organization music creative
|       `-- Step 18
|           Reason: Look for Renato Paris ICMP or BIMM or Tech Music School.
|           Query:  "Renato Paris" ICMP BIMM Tech Music School Berklee Trinity
|-- Step 8
|   Reason: Try finding the exact interview piece via different phrasing.
|   Query:  "my style is eclectic" interview "dangerous" Instagram Facebook
|-- Step 9
|   Reason: Search for specific quoted phrases from a November 2019 artist interview.
|   Query:  "social media is dangerous" interview 2019 musician eclectic
`-- Step 14
    Reason: Try finding interview with specific phrases about connecting via social media.
    Query:  "best way to connect" Instagram Twitter interview eclectic 2019
```

Figure 6: Case-study query trajectory for Claude Opus 4.7.