



retain. We therefore introduce Piika, a search agent equipped with three tools—search, read\_search\_results, and read\_document—that separate retrieval, result browsing, and document reading. Through this interface, Piika 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., 2026b; Meng et al., 2026), forcing the agent to terminate under a hard wall-clock time budget.

On BrowseComp-Plus (BCP) (Chen et al., 2026b), under time-budget steering, Piika with GPT-5.5 and tuned BM25 achieves 83.1% answer accuracy and 94.7% surfaced evidence recall, outperforming released dense-retriever search agents. Ablations show that BM25 tuning improves answer accuracy by 18.0 percentage points and surfaced evidence recall by 11.1 percentage points over the default BM25 setting, while increasing retrieval depth improves surfaced evidence recall by 25.3 percentage points over shallow retrieval.

Finally, Piika 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 Piika 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 rankings. Our contributions are as follows:

- We introduce Piika, 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  $3.3 \times - 10 \times$  lower running cost.

## 2 Related Work

**Deep Research.** Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) grounds LLM answer generation in external documents, but deep research, a complex information-seeking task, often requires more than a single retrieval step. It involves 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; Tongyi DeepResearch 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., 2025). As these systems become more capable, their evaluation has 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., 2026b). These benchmarks make retrieval and evidence interaction central to agent effectiveness, raising the question of how much effectiveness depends on the retrieval component itself.

**Retrieval Systems for Deep Research.** Search agents often use dense or learned sparse retrievers to generate initial candidate rankings, followed by rerankers to improve ranking quality (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., 2026a). By contrast, lexical methods such as BM25 are commonly used as baselines, and their weaker end-to-end effectiveness 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 (Sharifmoghaddam 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 Piika

Piika is a search agent for isolating agent–retriever interaction. The LLM agent runs a ReAct loop (Yao et al., 2023), 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 rankings 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.

Piika 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, Piika 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.1: 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 the cached ranking identified by `search_id` using rank-based pagination, exposing document IDs and excerpts on each page 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, Piika records four document sets:

- $D_{\text{surfaced}}$ : documents returned by search.
- $D_{\text{previewed}}$ : documents whose excerpts are shown through search or `read_search_results`.
- $D_{\text{opened}}$ : documents read via `read_document`.
- $D_{\text{cited}}$ : 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, Piika uses a two-stage time-budget policy. At  $0.7T$ , we inject a message to steer 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 various dense retrievers and BM25, measuring both retrieval effectiveness and final answer quality.

### 4.1 Evaluation Settings

**Datasets.** We evaluate Piika on BCP (Chen et al., 2026b), 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.

To evaluate whether our findings extend beyond BCP, we additionally evaluate Piika on TREC DL19 and DL20 and report the results in Section 7.

**Baselines.** We compare Piika with several representative BCP baseline systems. The released systems pair o3 and GPT-5 with either Qwen3-Embedding-8B (Zhang et al., 2025) or BM25. We also report the baseline from the work of Meng et al. (2026), which also uses the Qwen3-Embedding-8B retriever but replaces the LLM with GPT-5.2. Finally, we report the numbers of AgentIR (Chen

et al., 2026a), a reasoning-intensive retriever trained for deep research.

Unlike Piika’s three-tool interface, the baseline agents use a single search tool that directly exposes all top-5 retrieved results to the agent. Their surfaced and previewed document sets therefore coincide. We report these results as surfaced recall, but they also represent previewed recall and can be compared with Piika’s previewed recall.

**Metrics.** Following BCP, we report both answer-quality 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 final answer is semantically equivalent to the correct answer. We use GPT-5.3-Codex as the judge model; although it differs from BCP’s judge model, it reproduces the BCP evaluation outcome in Section 6. Appendix A.4 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:

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

All retrieval-recall metrics are computed from the logged document sets against the same fixed qrels and are therefore independent of the LLM judge used for answer evaluation.

For TREC DL19 and DL20, we additionally report nDCG@30 and Recall@100. We select these cutoffs to reflect retrieval in agentic-search settings. Agents can inspect more retrieved documents than conventional shallow cutoffs capture, whereas Recall@1000 represents a candidate pool that is too large to inspect effectively. Recall@100 therefore measures whether multiple searches retrieve a broad but practically accessible candidate set, while nDCG@30 measures whether relevant documents are prioritized near the top of the final ranking.

## 4.2 Piika 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 Piika runs in Table 1. Further details on BM25 tuning are provided in Section 6.

**Models.** We evaluate Piika using a diverse set of frontier LLMs with strong 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 Piika’s design remains effective across different LLM families and scales.

**Agent Harness.** We build Piika 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 replacing its system prompts and built-in tools with 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, Piika 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-Embedding-8B 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, Piika with GPT-5.5 in row (m) achieves the highest accuracy, at 83.1%. This is also the highest accuracy in Table 1,

| LLM                              | Retriever          | Answer Quality |        |              | Surfaced Recall |             | Previewed Recall |             | Behavior Recall |      |
|----------------------------------|--------------------|----------------|--------|--------------|-----------------|-------------|------------------|-------------|-----------------|------|
|                                  |                    | Acc.           | Calib. | Cost (\$)    | Evi.            | Gold        | Evi.             | Gold        | Evi.            | Gold |
| Chen et al. (2026b) <sup>†</sup> |                    |                |        |              |                 |             |                  |             |                 |      |
| (a) o3                           | BM25               | 50.8           | 39.1   | 836.35       | 56.6            | 61.7        | -                | -           | -               | -    |
| (b) o3                           | Qwen3-Embedding-8B | 66.3           | 32.7   | 740.79       | 73.2            | 76.3        | -                | -           | -               | -    |
| (c) GPT-5                        | BM25               | 58.3           | 13.5   | 400.36       | 61.7            | 66.5        | -                | -           | -               | -    |
| (d) GPT-5                        | Qwen3-Embedding-8B | 73.0           | 9.7    | 360.71       | 79.0            | <b>81.3</b> | -                | -           | -               | -    |
| Meng et al. (2026) <sup>‡</sup>  |                    |                |        |              |                 |             |                  |             |                 |      |
| (e) GPT-5.2                      | Qwen3-Embedding-8B | 45.1           | -      | 1 ~ 2k       | -               | 74.7        | -                | -           | -               | -    |
| Chen et al. (2026a) <sup>‡</sup> |                    |                |        |              |                 |             |                  |             |                 |      |
| (f) Tongyi-DR                    | AgentIR-4B         | 68.1           | -      | -            | <b>79.2</b>     | -           | -                | -           | -               | -    |
| Piika (ours)                     |                    |                |        |              |                 |             |                  |             |                 |      |
| (g) Claude Haiku 4.5             | BM25               | 54.8           | 17.3   | 193.48       | 94.1            | 95.4        | 58.2             | 60.1        | 40.9            | 46.3 |
| (h) Claude Opus 4.7              |                    | 69.8           | 10.2   | 246.62       | 81.2            | 86.8        | 43.3             | 52.5        | 30.4            | 43.0 |
| (i) GPT-5                        |                    | 74.6           | 7.2    | 94.87        | 90.5            | 93.8        | 62.7             | 70.0        | 45.7            | 56.2 |
| (j) GPT-5.2                      |                    | 70.5           | 6.2    | 122.23       | 89.9            | 92.8        | 60.5             | 67.3        | 44.8            | 54.5 |
| (k) GPT-5.4-mini                 |                    | 68.1           | 13.7   | 86.15        | 91.9            | 94.1        | 60.1             | 65.4        | 43.1            | 52.1 |
| (l) GPT-5.4                      |                    | 73.3           | 9.2    | 175.54       | 93.8            | 95.3        | 70.3             | 66.9        | 51.8            | 58.1 |
| (m) GPT-5.5                      |                    | <b>83.1</b>    | 15.7   | 291.61       | <b>94.7</b>     | 94.4        | <b>73.6</b>      | <b>72.9</b> | 58.9            | 56.1 |
| (n) DeepSeek-V4-Flash            |                    | 68.1           | 15.5   | <b>28.88</b> | 94.5            | <b>95.7</b> | 67.9             | 69.9        | 55.2            | 60.6 |
| (o) DeepSeek-V4-Pro              |                    | 71.4           | 7.0    | 55.13        | 91.3            | 92.5        | 60.0             | 63.0        | 45.4            | 50.6 |

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

<sup>†</sup> Results from Chen et al. (2026b) are obtained by evaluating their released runs using our own judge.

<sup>‡</sup> Results from Meng et al. (2026) and Chen et al. (2026a) are quoted from the original papers.

further showing that a BM25-based search agent can achieve highly competitive answer accuracy 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-Embedding-8B in row (e), achieves only 45.1% accuracy. In contrast, Piika 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 favorable accuracy–cost trade-off:

DeepSeek-V4-Flash achieves 68.1% accuracy with only \$28.88 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 analyze their behavioral differences in Appendix D.

**Retrieval Behavior.** Table 1 compares retrieval behavior across systems. For surfaced recall, all Piika 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 79.2 and 81.3 for the dense-retriever baselines of rows (f) and (d).

Under the same GPT-5 + BM25 setting, Piika also exposes more evidence to the agent. Compared with the released baseline in row (c), Piika in row (i) improves previewed recall of gold documents from 66.5 to 70.0, while maintaining similar previewed evidence-document recall. With GPT-5.5, Piika in row (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 rankings and allocate context to the right evidence.

**Cost Analysis.** Prior work on BCP has shown that deep research experiments can be expensive (Chen et al., 2026b; 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.

Piika substantially lowers this cost while preserving competitive answer accuracy. Under the same GPT-5 + BM25 setting, Piika reduces the cost from \$400.36 in row (c) to \$94.87 in row (i). When paired with more expensive LLMs, Piika remains cost-efficient. For example, GPT-5.5 + BM25 in row (m) achieves the highest accuracy at a cost of \$291.61, which is still lower than the GPT-5 baselines in rows (c) and (d). Similarly, Piika with GPT-5.2 costs \$122.23, far below roughly \$1000–\$2000 reported for the GPT-5.2 record in row (e).

These cost reductions matter because deep research studies depend on repeated evaluation. By lowering the cost of each full benchmark run, Piika 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 and DeepSeek-V4-Flash variants, most Piika 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, Piika uses only 15.2. In contrast, the GPT-5.2 + Qwen3-Embedding-8B system reported by Meng et al. (2026) uses a dense retriever but requires substantially more tool calls, 73.8 per query, while not

| System                     | Total | Search | Read | Browse |
|----------------------------|-------|--------|------|--------|
| <b>Chen et al. (2026b)</b> |       |        |      |        |
| o3 + BM25                  | 25.9  | 25.9   | –    | –      |
| o3 + Qwen3                 | 24.0  | 24.0   | –    | –      |
| GPT-5 + BM25               | 23.2  | 23.2   | –    | –      |
| GPT-5 + Qwen3              | 21.7  | 21.7   | –    | –      |
| <b>Chen et al. (2026a)</b> |       |        |      |        |
| Tongyi-DR + AgentIR-4B     | 27.9  | 24.5   | 3.4  | –      |
| <b>Meng et al. (2026)</b>  |       |        |      |        |
| GPT-5.2 + Qwen3            | 73.8  | 73.8   | –    | –      |
| <b>Piika (ours)</b>        |       |        |      |        |
| 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. Qwen3 denotes Qwen3-Embedding-8B.

achieving better answer accuracy than Piika. This indicates that replacing dense retrievers with BM25 does not necessarily increase the number of tool calls, while maintaining competitive retrieval effectiveness and answer accuracy, as shown in Table 1.

## 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 86.2%. 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 Piika. 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.** BM25 defaults target shorter documents, but documents in BCP are long and noisy. We test whether Piika’s gains depend on retuning BM25 for this setting.

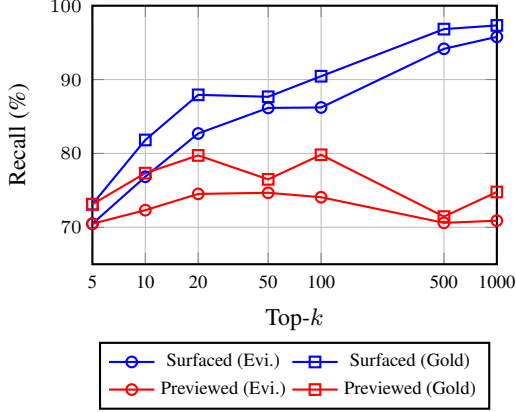


Figure 2: Effect of the number of documents ( $k$ ) returned by the search tool in Piika with GPT-5.4.

We sample 100 queries from BCP and perform a grid search over  $k_1$  and  $b$ . Figure 3 shows that Anserini’s default setting ( $k_1 = 0.9, b = 0.4$ ) lies in a region with low recall. 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 test the best grid-search setting,  $k_1 = 16$  and  $b = 1$ , and observe similar results. Tuning raises accuracy from 64.0% to 82.0% (81.0% 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  $\sim 2k$  tokens, and the 90th percentile has  $\sim 14k$  tokens (cf. Figure 4 in Chen et al. (2026b)). 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).

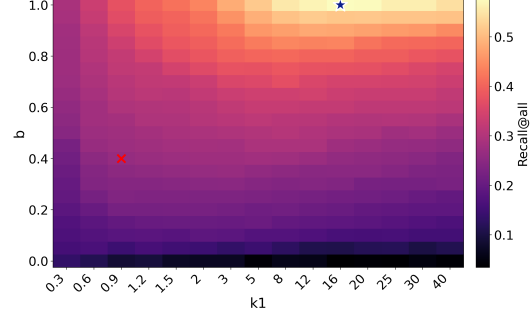


Figure 3: Grid search results of BM25 tuning over different  $k_1$  and  $b$  combinations.  $\star$  indicates the configuration with the highest recall, while  $\times$  marks Anserini’s default setting: ( $k_1 = 0.9, b = 0.4$ ).

**Cost Efficiency.** To better understand which designs in Piika 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 Piika; (2) a maximum of 100 iterations, which aligns with the baseline setting (Chen et al., 2026b; 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 \$24.88 to \$21.74, while substantially increasing the timeout to 3600 seconds does not dramatically increase the cost.

Regarding prefix cache, Piika inherits Pi’s agentic loop, which structures the interaction history 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 81% to 85% 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 Piika’s overall cost efficiency.

**Reproducing BCP.** We reproduce the released BCP search-only setup with Piika, keeping the search-only tool interface, top-5 retrieval depth,

|                                | Default    | Tuned   | Tuned     | Tuned   |
|--------------------------------|------------|---------|-----------|---------|
| BM25 ( $k_1, b$ )              | (0.9, 0.4) | (25, 1) | (25, 1)   | (25, 1) |
| Execution limit                | 300s       | 300s    | 100 iters | 3600s   |
| <i>Answer quality</i>          |            |         |           |         |
| Overall (%)                    | 64.0       | 82.0    | 76.0      | 83.0    |
| Completed (%)                  | 66.7       | 83.7    | 92.7      | 83.0    |
| <i>Retrieval effectiveness</i> |            |         |           |         |
| 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    |
| <i>Behavioral diagnosis</i>    |            |         |           |         |
| Avg. calls                     | 22.6       | 23.1    | 27.3      | 27.1    |
| Med. total tokens (k)          | 281.3      | 271.1   | 263.9     | 218.1   |
| Med. cached tokens (k)         | 228.7      | 228.0   | 215.6     | 186.1   |
| Cost (\$)                      | 24.11      | 21.74   | 24.88     | 26.33   |

Table 3: Effectiveness, cost, and tool-use comparison between default/tuned BM25 under different termination policies tested by Piika with GPT-5.4.

and default BM25 parameters fixed. As shown in the first two columns of Table 4, Piika achieves comparable answer accuracy to BCP, 54.3% versus 58.3%, while reducing cost from \$400.36 to \$140.58. Surfaced recall is lower, 49.9% versus 61.7%. The similar answer accuracy suggests that Piika provides a cheaper reproduction of the BCP setup, while the recall difference may stem from differences in prompts and search-result formats.

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 the 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 \$140.42 to \$94.87. These results suggest that Piika improves cost efficiency by letting the agent explore cached rankings, though further gains may require stronger support for selecting evidence from that ranking.

## 7 Effectiveness Generalization

To examine whether our findings extend beyond BCP, we evaluate Piika on TREC DL19 (Craswell et al., 2020) and DL20 (Craswell et al., 2021). We evaluate ranking quality using nDCG@30 and relevant-document coverage using Recall@100.

Unlike BCP, TREC DL directly evaluates retrieval rather than question answering. We therefore replace the BCP question-answering prompt

|                                | BCP<br>Chen et al. (2026b) | Partial<br>Piika | Partial<br>Piika | Full<br>Piika |
|--------------------------------|----------------------------|------------------|------------------|---------------|
| Tools available                | 1                          | 1                | 1                | 3             |
| Tuned BM25                     | –                          | –                | ✓                | ✓             |
| Retrieval Depth                | 5                          | 5                | 5                | 1k            |
| Agent Harness                  | Custom                     | Pi               | Pi               | Pi            |
| BM25 ( $k_1, b$ )              | (.9,.4)                    | (.9,.4)          | (25,1)           | (25,1)        |
| <i>Answer quality</i>          |                            |                  |                  |               |
| Overall (%)                    | 58.3                       | 54.3             | 70.7             | 74.6          |
| <i>Retrieval effectiveness</i> |                            |                  |                  |               |
| Surfaced (%)                   | 61.7                       | 49.9             | 69.1             | 90.5          |
| Previewed (%)                  | –                          | –                | –                | 62.7          |
| Behavior (%)                   | –                          | –                | –                | 45.7          |
| Cost (\$)                      | 400.36                     | 140.58           | 140.42           | 94.87         |

Table 4: Ablation of Piika components on BCP. All Piika variants use GPT-5 to match BCP settings. BCP results are from Chen et al. (2026b).

with a ranking prompt that asks the agent to return a ranked list of document identifiers. The rest of the search workflow remains unchanged. The exact prompt is provided in Appendix A.4.

From the same search trajectory, we construct two output rankings. *Subquery RRF* aggregates the BM25 ranked lists from all search calls using reciprocal rank fusion, yielding a top-100 candidate set without requiring the LLM to generate a long list of document identifiers. *Listwise ranking* instead uses the agent’s final response to select and order the top-30 documents based on the evidence accumulated during the same trajectory. The distinction therefore lies in output-ranking construction, rather than in the underlying search process. As a non-agentic baseline, we use Qwen3-Embedding-8B as a single-query dense retriever, which retrieves directly from the original query without query reformulation or iterative search.

Table 5 illustrates the complementary roles of these two output rankings. Subquery RRF achieves Recall@100 scores of 0.55 on DL19 and 0.61 on DL20, close to 0.58 and 0.62 for the dense retriever. Meanwhile, listwise ranking achieves nDCG@30 scores of 0.71 and 0.72, compared with 0.73 and 0.70 for the dense retriever. Thus, the same sequence of lexical searches supports both broad candidate coverage through result aggregation and competitive top-30 relevance through the agent’s final ranking.

| System                                         | Ranking              | DL19    |            | DL20    |            |
|------------------------------------------------|----------------------|---------|------------|---------|------------|
|                                                |                      | nDCG@30 | Recall@100 | nDCG@30 | Recall@100 |
| Qwen3-Embedding-8B<br>(single-query retrieval) | Embedding similarity | 0.73    | 0.58       | 0.70    | 0.62       |
| Piika + GPT-5.5                                | Subquery RRF         | 0.61    | 0.55       | 0.63    | 0.61       |
|                                                | Listwise ranking     | 0.71    | -          | 0.72    | -          |

Table 5: TREC DL19/DL20 results. Piika’s two output rankings are constructed from the same search trajectory: subquery RRF aggregates search results for candidate coverage, while listwise ranking prioritizes relevance in the top 30. We omit Recall@100 for listwise ranking because its output is limited to 30 documents.

## 8 Discussion

Recent work by Cao et al. (2026) and Li et al. (2026) shows that coding agents can achieve competitive answer accuracy 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 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). Therefore, we report these systems as reference points in Figure 1, rather than as direct comparisons in Table 1.

Nevertheless, their systems study how agents identify useful context in locally available documents. Our results complement this setting: with proper configuration and sufficient retrieval depth, BM25 can provide a high-recall candidate set for the agent to inspect. Appendix B reports a preliminary BM25–dense comparison under the same agent setting.

## 9 Conclusion

We ask: *Does a lexical retriever suffice as LLMs become more capable within 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. Results on TREC DL19/DL20 further suggest that these findings extend beyond BCP. 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 pushing the boundaries of deep research systems, the most useful signal is where Piika fails. It often surfaces the right evidence documents, but the agent 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 identify and focus its token budget on the right evidence in the available documents?* 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.

## 10 Limitations

Our study has several limitations. First, Piika relies on a relatively basic interface for navigating the cached rankings, resulting in lower previewed and behavior recall than surfaced recall. Second, our evaluation is limited to BCP and TREC DL19/DL20, leaving open the question of whether the agent harness generalizes to multilingual and domain-specific scenarios. Finally, although using a timeout as the termination policy simplifies execution control, queries of varying complexity may require different amounts of research time. A fixed time budget may fail to balance effectiveness, latency, and monetary cost, leading either to insufficient exploration for difficult queries or unnecessary computation for simpler ones.

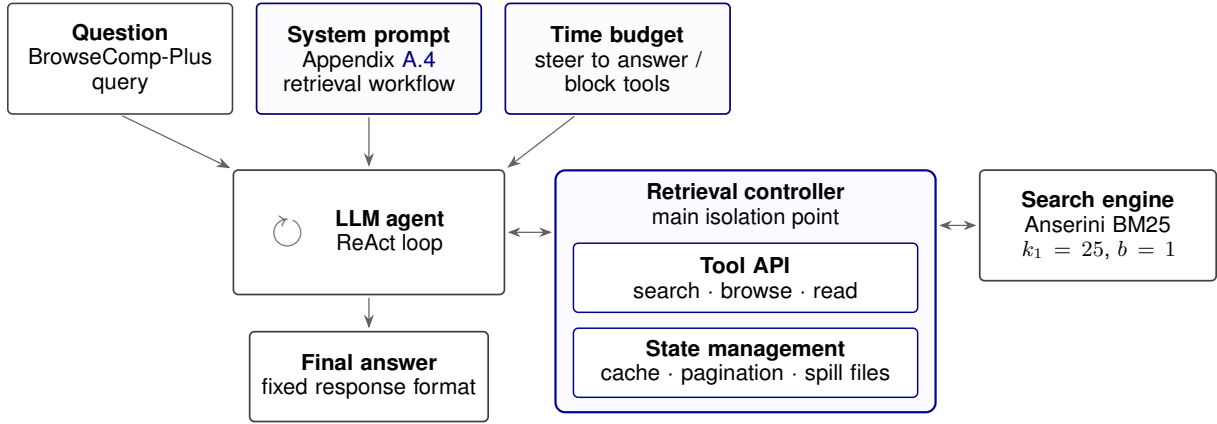
## Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Additional funding was provided by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project).

## 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](#). *arXiv:2603.20432*.
- Zijian Chen, Xueguang Ma, Shengyao Zhuang, Jimmy Lin, Akari Asai, and Victor Zhong. 2026a. [AgentIR: Reasoning-aware retrieval for deep research agents](#). *arXiv:2603.04384*.
- Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, and 2 others. 2026b. [BrowseComp-Plus: A fair and disentangled evaluation benchmark for deep search agents](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22349–22370, San Diego, California, United States. Association for Computational Linguistics.
- Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. [Overview of the TREC 2020 deep learning track](#). *arXiv:2102.07662*.
- Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. [Overview of the TREC 2019 deep learning track](#). *arXiv:2003.07820*.
- Tiansheng Hu, Yilun Zhao, Canyu Zhang, Arman Cohan, and Chen Zhao. 2026. [SAGE: Benchmarking and improving retrieval for deep research agents](#). *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, Wenhui Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. 2026. [Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction](#). *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*. *arXiv:2602.21456*.
- Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiuxi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. [ToolRL: Reward is all tool learning needs](#). In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*.
- Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muenighoff, 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](#). *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](#). *arXiv:2402.03300*.
- Sahel Sharifymoghaddam and Jimmy Lin. 2026. [Rerank before you reason: Analyzing reranking tradeoffs through effective token cost in deep search agents](#). *arXiv:2601.14224*.
- Zhengliang Shi, Yiqun Chen, Haitao Li, Weiwei Sun, Shiyu Ni, Yougang Lyu, Run-Ze Fan, Bowen Jin, Yixuan Weng, Minjun Zhu, Qiuqie Xie, Xinyu Guo, Qu Yang, Jiayi Wu, Jujia Zhao, Xiangqiang Tang, Xinbei Ma, Cunxiang Wang, Jiaxin Mao, and 7 others. 2025. [Deep research: A systematic survey](#). *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](#). *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](#). *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*. *arXiv:2210.03629*.
- 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*, pages 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](#). *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](#). *arXiv:2504.19314*.



Blue components belong to Piika; the retrieval controller is the main isolation point between the agent and search engine.

Figure 4: System architecture of Piika. 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 Piika setup used in our experiments. We organize the setup by prompt, tool interface, backend configuration, and termination policy.

### A.1 Retrieval Tool Interface

Figure 5 summarizes the interfaces for `search`, `read_search_results`, and `read_document`. These tools retrieve documents, browse cached rankings, and read document chunks. Arguments marked `[default:...]` are optional. Return entries summarize the output contents rather than literal response-field names.

### A.2 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. The pi-search configuration schema currently supports several backend kinds, including Anserini-BM25 over TCP, Anserini-BM25 over stdio, HTTP-JSON, and a mock backend for testing. The detailed configuration format for each backend is provided in the project repository.

**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.3 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  $\text{floor}(0.7 \times \text{TIMEOUT\_SECONDS} \times 1000)$  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`.

| search                                                                                                                     |
|----------------------------------------------------------------------------------------------------------------------------|
| Send a lexical query to the backend. BM25 uses query_mode=plain, caches up to 1000 hits, and initially displays ranks 1-5. |
| Arguments:                                                                                                                 |
| reason Brief rationale. [required]                                                                                         |
| query Raw query string. [required]                                                                                         |
| Returns:                                                                                                                   |
| search_id Cached ranking identifier.                                                                                       |
| Page Formatted first page.                                                                                                 |
| Doc IDs Cached and displayed IDs.                                                                                          |
| Metadata Timing and truncation.                                                                                            |

| read_search_results                                                         |
|-----------------------------------------------------------------------------|
| Browse a page of cached search results without issuing a new backend query. |
| Arguments:                                                                  |
| reason Brief rationale. [required]                                          |
| search_id ID from search. [required]                                        |
| offset Starting rank, 1-indexed. [default: 6]                               |
| limit Maximum hits to display. [default: 10]                                |
| Returns:                                                                    |
| Page Formatted result page.                                                 |
| Doc IDs Displayed document IDs.                                             |
| next_offset Next rank, if hits remain.                                      |
| Metadata Truncation details if the page exceeds output limits.              |

| read_document                                       |
|-----------------------------------------------------|
| Read a backend document in line-based chunks.       |
| Arguments:                                          |
| reason Brief rationale. [required]                  |
| docid Document identifier. [required]               |
| offset Starting line, 1-indexed. [default: 1]       |
| limit Maximum lines to read. [default: 200]         |
| Returns (when found):                               |
| Text Formatted document lines.                      |
| Lines Returned range and document line count.       |
| Truncation Backend truncation flag.                 |
| next_offset Next line, when available.              |
| Metadata Timing; output truncation when applicable. |

Figure 5: Retrieval tool interfaces.

## A.4 Prompts

**Task Prompts.** Figures 6 and 7 show prompts for BCP question answering and TREC DL19/DL20 ranking, respectively. Placeholders in curly braces, such as {Question}, are replaced at runtime.

**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 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}
```

## Question-Answering 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}

Figure 6: Prompt used for BCP question answering.

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. Figure 8 shows the judge prompt.

## B Controlled Dense-Retrieval Study

We conduct a preliminary controlled comparison on 300 sampled BCP queries to examine whether replacing BM25 with dense retrieval improves answer accuracy and surfaced recall under the same agent architecture. All variants use GPT-5.4 with the same task prompt, tool interface, time-budget steering policy, and evaluation protocol. The only changed component is the retrieval backend. We compare lexical retrieval with tuned BM25 against dense retrieval with Qwen3-Embedding-0.6B.

Lexical retrieval achieves 95.67% surfaced recall and 82.00% end-to-end accuracy. Dense retrieval

### TREC Ranking Prompt

You are a research and retrieval agent 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. Satisfy every requested output below; use the same research pass for all outputs.
9. As soon as you have enough evidence and candidates, stop using tools and answer in plain assistant text.

Ranked-list output:

- Return exactly 30 unique document ids. Continue searching until you have at least 30 plausible candidates; do not return fewer than 30.

Your final response must contain exactly these sections in this order:

Ranked List:

1. {docid}
2. {docid}
3. {docid}

Do not include any text before, after, or between those sections beyond the requested fields and ranked document lines.

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.

Question: {query}

Figure 7: TREC DL19/DL20 prompt for producing the agent’s final top-30 listwise ranking.

achieves 88.00% surfaced recall and 67.00% accuracy. Thus, replacing BM25 with this dense retriever doesn’t improve answer accuracy in this setting. These findings reinforce our claim that BM25 has been underestimated in prior agentic-search settings. With an appropriate retriever configuration and an agent–tool interface supporting iterative querying, deeper result access, and aggregation across searches, a BM25-based search agent can construct a strong candidate pool supporting competitive answer accuracy. Our claim is not that BM25 universally outperforms dense retrieval, but that agentic-search effectiveness depends on the interaction among retriever configuration, tool-interface design, and downstream agent behavior, rather than on retriever strength alone.

### 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}

Figure 8: Gold-answer judge prompt for evaluating answer correctness on BCP.

## C Model Token Pricing Used in Benchmark Runs

Table 6 reports the token pricing used by our benchmark runs. 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      |

Table 6: Token pricing used during BCP execution.

## D 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 answer-accuracy 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 these 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 9 and 10 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 9, 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 10 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 9: 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: Forli Italy population history spelling.
| Query: Forli Wikipedia population history name origin
|
|-- Step 12
| Reason: Look for Forli Wikipedia article in the corpus.
| Query: Forli city Italy Emilia-Romagna
|
|-- Step 16
| Reason: Try different angle---artist from Forli grew up there.
| Query: "grew up in Forli" musician artist
|
|-- Step 17
| Reason: Try searching for transcription of the interview content with Italian connection.
| Query: Renato Paris singer Italian Forli 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 10: Case-study query trajectory for Claude Opus 4.7.