Hawkeye: Inspecting Agentic Search Trajectories

Crystina Zhang[1*] **Zigeng Xu**[2*] **Xinyu Shi**[1] **Raphael Tang**[3,4] **Nandan Thakur**[1] **Xueguang Ma**[1]
Yao Lu[3] **Jimmy Lin**[1]

[1]University of Waterloo [2]The Chinese University of Hong Kong, Shenzhen
[3]University College London [4]Microsoft

Abstract

Agentic search systems answer complex questions by iteratively issuing subqueries, retrieving evidence, and deciding whether and how to continue the search. However, existing evaluations often reduce this process to final accuracy or the number of search calls, obscuring whether an agent is exploring useful topics, reusing prior evidence, or wasting context on redundant documents. We present HAWKEYE, a visual analytics tool for inspecting agentic search trajectories at both the per-question and dataset levels. HAWKEYE records each question, LLM-issued subquery, and retrieved document, and visualizes three complementary aspects: *topic shift* across subqueries, *query provenance* attributing query tokens to the original question, retrieved context, LLM parametric memory, or previous queries, and *document novelty* measuring how many retrieved documents in each round are new rather than repeated from earlier rounds. Using HAWKEYE to analyze BrowseComp-Plus runs, we identify behaviors that are difficult to observe from aggregate metrics alone, including repeated retrieval of previously seen documents and sparsely use newly retrieved context in query formulation. These observations show how trajectory visualization can help diagnose agentic search failures and suggest directions for improving agentic search systems.

1 Introduction

Agentic search allows language models to answer complex questions by iteratively decomposing them into subqueries, retrieving external evidence, reasoning over the retrieved content, and deciding whether to search again or produce a final answer. (Lewis et al., 2020; Wei et al., 2025; Chen et al., 2025a; Asai et al., 2024) This paradigm has become a common way to extend language models beyond their parametric knowledge, but also add complexity in inspecting the information seeking process. A final answer and a scalar accuracy score reveal little about whether the model asked useful subqueries, whether the retriever returned informative documents, or whether later queries actually made use of the evidence collected earlier.

Existing evaluations often summarize an agentic search run by outcome-level statistics such as accuracy, number of search calls, or total context length. (Chen et al., 2025b; Hu et al., 2026) These signals are useful but incomplete: two trajectories with the same number of search rounds can differ substantially in whether the model is exploring new topics, repeating earlier queries, or accumulating redundant documents. For failed questions, it is therefore hard to tell whether the bottleneck comes from query formulation, retrieval quality, evidence utilization, or any other factors. For correctly answered questions, it is also hard to identify whether all search calls have been necessary and helpful.

To make this process easily inspectable to all practitioners, we present HAWKEYE, an agentic search trajectory visualizer. HAWKEYE records each question, LLM-issued subquery, and retrieved documents, rendering the resulting trajectory at both the per-question and dataset levels. The tool is designed to expose three complementary aspects: *topic shift*, which shows how the semantic topics of subqueries change over time; *query provenance*, which attributes query tokens to the original question, retrieved context, LLM parametric memory, or previously issued queries; and *document novelty*, which measures how many retrieved documents in each round are new rather than repeated from earlier rounds. Together, these aspects help analysts move beyond counting search calls toward understanding how the agent searches and whether each round contributes new information.

We use HAWKEYE to inspect agentic search runs on BrowseComp-Plus and observe patterns that are

[*]Equal contribution.

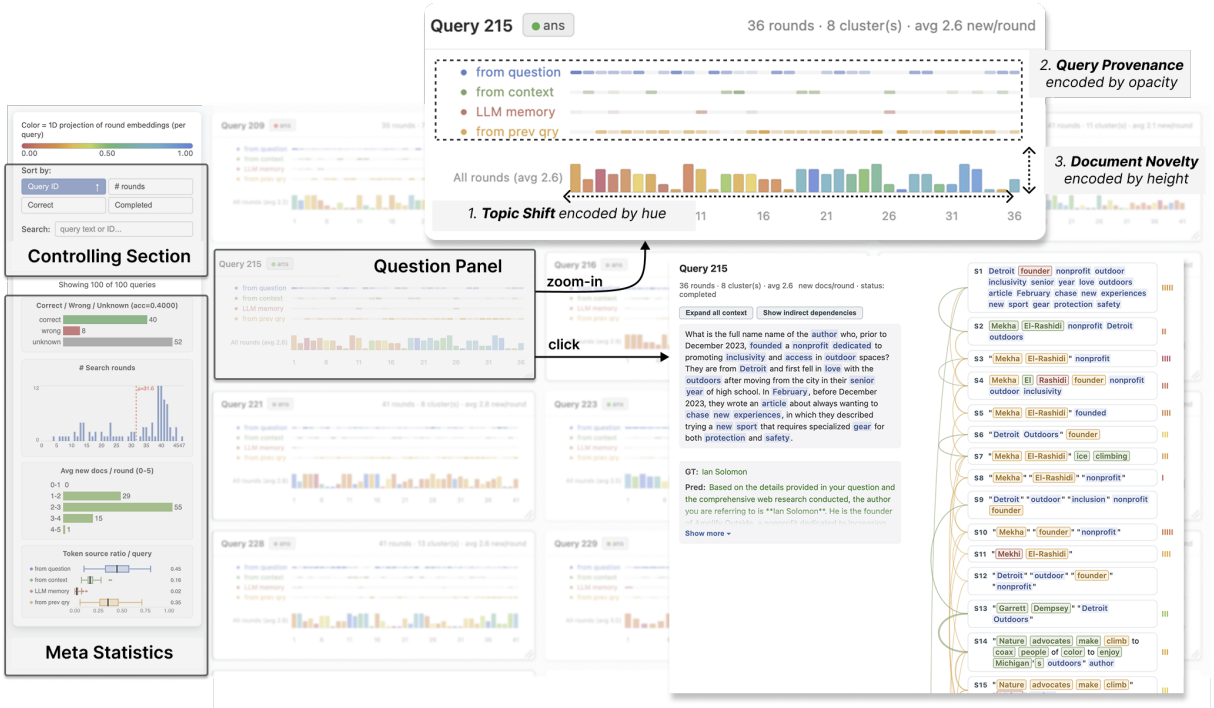

Figure 1: HAWKEYE interface, which comprises three main parts: *Controlling section* for adjusting query layouts, *Meta statistics* for a dataset-level overview, and *Question panels* for the search trajectory per question.

difficult to see from aggregate accuracy alone. For example, many trajectories repeatedly retrieve documents that have already been shown to the model, and removing such duplicate documents improves accuracy across the retrievers we evaluate. We also find that later queries draw more heavily from the original user-question and previous LLM-issued queries than from retrieved context, suggesting that agents only sparsely use newly retrieved context in query formulation. These observations illustrate how trajectory-level visualization can diagnose failure modes and suggest concrete directions for improving agentic search systems.

2 Usage

The use of HAWKEYE is simple and can be demonstrated in the code snippet below:

```
distance = distances.Cosine(encoder)
with Hawkeye(output_path) as hawkeye:
    for question in questions:
        tracker = hawkeye.track(
            question, distance, *config
        ) as tracker:
        for query, doc in stream(
            question):
            tracker.log(query, doc)
        hawkeye.add(tracker)
```

A visualization interface will be rendered when exiting the HAWKEYE context manager, as shown in Figure 1, which will soon be introduced in the following section. The `distances.Cosine` class provide one of the distance metrics HAWKEYE supports for pairwise query distance computation. More implementation details and visualization efficiency will be provided in Section 4 and 5.

3 Overview

HAWKEYE interface in Figure 1 comprises three main parts: *Controlling section* that adjusts query layouts, *Meta statistics* that provides a dataset-level overview, and *Question panels* that characterize the search trajectory of each question.

3.1 Controlling Sections

The controlling section exposes the widgets that control the order of question panels and the quick-lookup filter. Specifically, *Sort by* supports ordering question panels by question identifiers, number of search rounds, correctness, or completion status, in either ascending or descending direction; this offers fast access to the slow-converging or incorrectly-answered trajectories that tend to be the most informative for analysis. The *search* box allows users to narrow the displayed panels to questions whose query text or identifier matches a substring, or whose retrieved documents intersect a chosen relevance tag (e.g. gold, evidence), supporting targeted inspection of subsets of interest.

3.2 Question Panel

This serves as the major analysis tool in HAWKEYE, which visualizes three complementary aspects given each logged question and its search trajectory, along with a detailed question page for further inspection. We hereby quicklly brief the functional design of each component, and leave implementation details in Section 4.

Topics Shift. This aspect aims to show how the topics of LLM-issued queries shift along the search trajectory. Specifically, we color-code the query topic similarity so that queries with similar topics are rendered in a similar color, and vice versa. (Section 4.1)

Query Provenance. This aspect explains the provenance of each LLM-issued query by attributing each query token to one of four sources: the original question, retrieved context, LLM parametric memory, or previous queries. The resulting query provenance is thus represented as four ratios, each measuring the percentage of query tokens attributed to one source. In the panel, these ratios are shown using four colors, where higher values correspond to more saturated colors. This visualization makes it easy to see how different sources of information flow into the reformulated queries over the search trajectory. (Section 4.2)

Document Novelty. This aspect measures how many *new* documents are introduced by each round of search, which is encoded using the bar height: taller bars indicate that the current query retrieves many documents that have not appeared in previous rounds, while shorter ones indicate that the query mostly retrieves documents already seen before and barely provides new information. This aims to distinguish informative search rounds from the redundant ones.

While above three aspects each have their own focus, they provide complementary information. For example, when *query provenance* shows that a query reuses existing vocabulary from previous queries, *document novelty* helps assess whether such query reuse is productive for finding new information. See Appendix A for a dataset-level view of the interplay between query provenance and document novelty.

Detailed Question Page To allow users to inspect a question and its retrieved documents in their original form, each question panel can be expanded into a dedicated detail page. The page is organized into two columns. The left column shows the question text, ground-truth answer, the agent's final response, and the evaluation result if available, which is always pinned to the top so that it remains visible while the user scrolls through the trajectory on the right column.

The right column lays out the rounds vertically as a *subquery dependency tree*: each round is rendered as a card containing the issued subquery, with its tokens highlighted in the four provenance colors introduced in Section 4.2; underneath each subquery, the retrieved documents are shown in collapsed form by default and can be expanded individually or collectively via a toggle in the controls.

Dependency arcs connect each token to its source, which also follows the color code of query provenance: orange arcs trace back to previous queries, green arcs to retrieved documents. Additionally, when hovered on a query panel, only "connected" query panels and corresponding query words from the user question would be highlighted. This allows to not only inspect the actual query token flow, but also identify "sub-trajectories" that lurks inside the entire search trajectory. See Appendix B and Figure 5 for a demo and discussion. These supports detailed inspection of how each trajectory unfolded.

3.3 Meta Statistics

The meta-statistics sidebar summarizes dataset-level behavior of the agent in four stacked panels, each of which is interactive and reflects the user's current selection. The first panel reports the accuracy breakdown of *correct*, *wrong*, and *unknown*[1] judge verdicts, along with the overall accuracy of the run;[2] The second panel shows histograms of the number of search rounds per question, which is a commonly reported metric to evaluate the effectiveness of agentic search.

The third and fourth panels tightly related to the features in question panel above: The third panel histograms the number of novel documents retrieved per round, providing a coarse summary of how informative or redundant each search round is. The fourth panel summarizes the distribution

[1] Unknown denotes questions for which the judge produced no verdict, either because the run did not terminate or because the judge itself abstained.

[2] Only applies if the accuracy information is available, not applicable if evaluation is not provided.

of *query provenance* across the four sources. For each question, we compute the average ratio of query tokens attributed to the original question, the retrieved context, LLM parametric memory, and previous queries (averaged over all search rounds); the resulting per-question ratios are then aggregated into four boxplots, one box per source. This shows at a glance *where the agent's queries tend to come from* across the entire dataset,

Among the four panels, the accuracy panel (first panel) also serve as a quick filter: clicking a bar applies a correctness filter that applies to all question panels and propagates to every other three meta statistics panels, so that the histogram and boxplot can be sliced to, for instance, contrast the provenance distribution of correctly- versus incorrectly-answered questions. See Appendix C and Figure 6 for an example and discussion.

4 Implementations

This section provides algorithmic details on visualizing topic shifts and query provenance.

4.1 Topics Shift

HAWKEYE color-codes topic similarity among subqueries in two steps: (1) compute pairwise distance among all subqueries, $D \in \mathbb{R}^{n \times n}$, and (2) learn a low-dimensional representation per subquery based on classic multidimensional scaling (MDS) (Borg and Groenen, 1997).

Pairwise Distances among Queries. HAWKEYE supports two types of distances to allow a user-decided trade-off between efficiency and accuracy:

1. *Lexicon-based distance* computes Jaccard distances between a pair of subqueries, where each query is tokenized into lexicons in the same way as described in Section 4.2. Since most queries are keyword-based, we observe that Jaccard distances can effectively capture topic similarity while achieving much lower rendering latency (more details in Section 5), enabling fast prototyping. However, it would fail in the presence of synonyms or language switches during the search trajectory.

2. *Embedding-based distance* computes cosine distances between a pair of subqueries based on user-specified embedding models supported in HuggingFace (Wolf et al., 2020). In contrast to the Jaccard distances above, this trade-off exchanges rendering latency for higher semantic similarity accuracy and is not limited by word-level mismatches.

Color Coding. To color-code the queries based on their pairwise distances, we first project each query to a 1-D coordinate by applying classical MDS to the pairwise distance matrix D, We fix the arbitrary sign of the first MDS coordinate by requiring its largest-magnitude entry to be positive, then min-max normalize the result to $t \in [0, 1]$. To ensure that the color changes could uniformly represent the semantic shift, we then map t to the hue channel in the OKLCh color space (Ottosson, 2020; W3C, 2024): each t becomes a hue $H = 20^\circ + t \cdot 230^\circ$ at fixed lightness $L = 0.70$ and chroma $C = 0.13$, traversing almost the full color spectrum from red → yellow → green → cyan → blue.[3] Holding L and C constant keeps perceived brightness and saturation identical across points, so hue alone carries the MDS coordinate; After hue is computed, each $(L, C\cos H, C\sin H)$ triple is converted from OKLab to sRGB for rendering.[4]

4.2 Query Provenance

Query provenance attempts to trace the source of each query token via lexical matching. Specifically, we tokenize the question, query, retrieved context, and prior queries, and compute the percentage of query tokens that exactly match tokens from each source, Tokens that cannot be matched to the question, context, or previous queries are categorized as LLM-memory tokens, which may reflect paraphrases, synonyms, language switches, factual substitutions, etc. Although LLM-memory is a rather broad attribution, we find that such tokens are rare in practice: only 1% of query tokens fall under this category on average on BrowseComp-Plus.[5]

Tokenization and Lexicon Tracing. The query token traceback is conducted on pure lexical space, based on normalized tokens using Porter stemmer (Porter, 1980), and after removing English stopwords to allow morphological variants of the same word to be aligned, a process that is

[3] Hue range from angle 20 degrees to 250 degrees, which is almost the full color spectrum but still leaves the two endpoints visually distinct. We intentionally avoid the violet-pink regions as it shows less perceptual uniformity.

[4] While there are more perceptually uniform color spaces, we choose OKLCh as it is computationally efficient and easy to implement.

[5] Figure 3 and Figure 4 show the average query provenance per source on BrowseComp-Plus while using Tongyi (Team, 2025b) and multiple retrievers.

commonly used in bag-of-words retrieval algorithms (Robertson and Zaragoza, 2009).

Specifically, each text span T — the question, a subquery, a retrieved document, or a previous subquery — is first segmented into alphanumeric tokens via a Unicode-aware regular expression, lowercased, filtered against an English stopword list, and finally reduced to its Porter stem. The output is a set of stemmed tokens $\mathcal{S}(T)$, which we use as the matching unit; purely numeric or symbolic tokens are passed through unchanged so that identifiers and figures remain matchable. While the lexical matching appears to be rigid, we find that over 99% of the query tokens are traceable to the original question, retrieved context, or previous queries in this way, suggesting that current LLMs tend to "copy-paste" available information instead of creating new query contents.

Given a search trajectory with rounds $k \in \{1, \ldots, K\}$, let $\mathcal{S}_q$ denote the question stems, $\mathcal{S}_{Q_k}$ the stems of the k-th subquery, and $\mathcal{S}_{D_k}$ the union of stems extracted from the documents retrieved at round k. For each round k, we partition $\mathcal{S}_{Q_k}$ into four disjoint sets based on their provenance:

1. *from the question:* Any query token that can be exactly matched to the question stems, formally: $\textsc{InQ}_k = \mathcal{S}_{Q_k} \cap \mathcal{S}_q$.

2. *from previous queries:* Any query token that can be exactly matched to tokens from any previous queries and not in $\textsc{InQ}_k$, formaly: $\textsc{InPrev}_k = (\mathcal{S}_{Q_k} \setminus \textsc{InQ}_k) \cap \bigcup_{j<k} \mathcal{S}_{Q_j}$.

3. *from the retrieved context:* Any query token that can be exactly matched to tokens from any previously retrieved context and not in $\textsc{InQ}_k \cup \textsc{InPrev}_k$, formaly: $\textsc{InCtx}_k = (\mathcal{S}_{Q_k} \setminus \textsc{InQ}_k \cup \textsc{InPrev}_k) \cap \bigcup_{j<k} \mathcal{S}_{D_j}$.

4. *from LLM memory:* all remaining stems form $\textsc{Ungrounded}_k$, as they have no surface support in the question, prior subqueries, or any document seen so far, and must therefore be formed through the agent's parametric knowledge.

The four ratios reported in the question panel for round k are then normalized by $|\mathcal{S}_{Q_k}|$. While all lexical matching happens in stem space, the visualizer renders the original surface forms, so the rendered labels remain human-readable while the underlying matching stays robust to morphological variations.

Relation to CTAR. The computation of query provenance is inspired by the Context-driven Term Adoption Rate (CTAR) proposed in Ning et al. (2026), which measures the fraction of *newly introduced query terms* that can be lexically traced to previously retrieved evidence. We will discuss that while both are lexical matching based, the two metrics are designed for different purposes, and the scores are not directly comparable.

First, CTAR is defined over newly introduced terms between two consecutive queries: it asks what fraction of those new terms can be traced back to the last round of retrieved context or to all accumulated context so far. In contrast, HAWKEYE attributes every query token, including tokens that have appeared in earlier queries, so its ratios are normalized by the total number of query tokens. Thus, the two metrics use different denominators and answer different questions: CTAR measures context adoption among new terms, while query provenance describes the information source of the entire query.

This difference also changes how repeated terms are interpreted. In CTAR, a term is considered new if it appears in q_{k+1} but not in q_k, which means a new term in q_{k+1} could have appeared in the previous query q_j where $0 \leq j < k$. This is desirable when the goal is to investigate query reformulation motivation *per-round*. In contrast, HAWKEYE aims to characterize the minimal information required to form each query token, which helps to inspect the smallest amount of external context required to form a query. Thus it worths to separate whether a token is newly adopted from retrieved context or has already seen in the "known vocabulary" through a previous query. In the later case, even if the token was originally retrieved from context, it would not require an additional search call to regather the same information as such information has already been presented to the LLM. This reflects our goal of characterizing the information flow of the agent at each round: once a term has appeared in a previous query, it no longer requires newly retrieved context to explain its reuse.

5 Visualization Efficiency

Figure 2 provides a breakdown of rendering latency per query when using different distance and embedding models, gathered on a single 140G-memory H200 and using batch size 256 if encoding is required. It shows that the latency bottleneck mainly

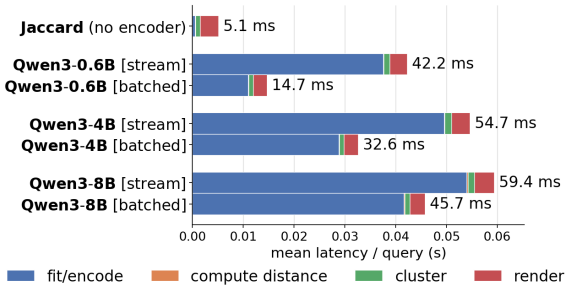

Figure 2: Breakdown of rendering latency per query. (Note that here the Qwen emedding models are used to compute pairwise query distance, and independent from the retrievers used in the agentic search process.)

comes from encoding queries into embeddings, whereas all other operations take minimal times (<5ms). We thus provide two modes for query encoding, *stream* and *batched*. In *stream* mode, queries are encoded while they are added to the session, which consumes less GPU memory and is better for running on the side. In *batched* mode, queries are buffered and encoded together when reaching a user-specified encoding batch to fully utilize GPU resources and maximize throughput.

The two modes target different user scenarios: The *stream* mode is to allow logging search trajectories on the fly. In such cases, query latency of around 50ms on a separate thread has minimal effect on overall runtime, while consuming minimal GPU resources. The *batched* mode is to allow visualizing the trajectory from cached files, in which case we hope the visualization could be completed at full speed.

6 Insights

The goal of building a visualization tool is to aid practitioners on spotting unexpected behaviors from the system, which hopefully leads to better understanding and further improvements. We hereby share some of the interesting findings made from Browsecomp-Plus as humble starts.

6.1 Experimental Setups

Evaluation Benchmark. Experiments under this section are based on BrowseComp-Plus, an evaluation benchmark for deep-research agents under a controlled retrieval environment derived from BrowseComp(Wei et al., 2025), It contains 830 queries and a fixed corpus of roughly 100K human-verified documents, allowing for running agentic search without access to live web search. We re-use the codebase provided by BrowseComp-Plus with modifications to cache and visualize the search trajectory, and add support to `Reason-ModernColBERT`.

Retriever	Entire Dataset		Context-Limit Subset	
	w/o filter	w/ filter	w/o filter	w/ filter
BM25	0.3422	0.4578	0.0000	0.2834
Qwen3-8B	0.5530	0.5988	0.0000	0.2905
R-MC	0.6663	0.6988	0.0000	0.3484

Table 1: Result comparison of filtering duplicate documents during agentic search, reporting accuracy on BrowseComp-Plus using Tongyi-30B as LLM and different retrievers. The context-limit subset contains questions whose no-filter baseline runs reach the maximum context length. "w/o filter" indicates original settings, "w/ filter" indicates to remove duplicate documents when returning retrieved documents to the LLM, and ask the LLM to regenerate the query if all documents are duplicates. Evaluations are based on Qwen-32B using the official evaluation script. "Qwen3-8B": Qwen3-Embedding-8B, "R-MC": Reason-ModernColBERT.

LLM and Retriever. Constrained by API and compute budget, all experiments use `Tongyi-DeepResearch-30B-A3B` (Team, 2025b) as the LLM backbone of the deep research agent, which has been shown to have superior deep-information seeking ability with an affordable model size. We use the default configuration in our runner: temperature 0.85, top-p 0.95, and presence penalty 1.1. Each search call returns top $k = 5$ documents, and each returned snippet is truncated to 512 tokens.

We experimented with multiple retrievers from lexical, embedding, and multi-vector families: BM25, `Qwen3-Embedding` (0.6B, 4B, 8B) (Team, 2025a), and `Reason-ModernColBERT` (Chaffin, 2025).

For BM25 and the Qwen3-Embedding retrievers, we use the pre-built indexes released by BrowseComp-Plus.[6] Specifically, BM25 uses the released Pyserini/Lucene index, while the Qwen3-Embedding retrievers use the released FAISS indexes with normalized embeddings. For `Reason-ModernColBERT`, we build a late-interaction index over the BrowseComp-Plus corpus using `lightonai/Reason-ModernColBERT`.[7]

[6] https://huggingface.co/datasets/Tevatron/browsecomp-plus-indexes

[7] https://huggingface.co/lightonai/Reason-ModernColBERT

`get_documents` In our experiments, we provide both `search` and `get_documents` tool usage to LLMs following (Chen et al., 2025b): `search` allows agents to receive truncated snippets from the top-ranked documents, whereas `get_document` allows agents to additionally fetch full document content given the document id. Compared to having only `search` tool, having `get_document` allows models to dynamically access the full context if they desire and thus avoid the failing cases where relevant information is not included in the first 512 tokens. On the other hand, this introduces the potential cost of longer context length. We choose to opt in `get_document` as it consistently provides higher accuracy on all our settings.

Remove duplicate documents. To study the effect of redundant documents, we run duplicate-document filtering ablations with Qwen3-8B and Reason-ModernColBERT in the `search` + `get_document` setting. The baseline returns retrieved documents as usual. We add a `filter` between the retriever and the LLM backbone, which removes documents that have already been shown earlier in the same question trajectory. Then the filtered documents are returned to the LLM. If no document remains after filtering, the LLM would reformulate the query and search again, which are told in system prompt.

6.2 Discussions

Duplicate Documents. In almost all search trajectories, we observe search rounds with duplicate documents that have been retrieved and presented to LLMs in previous rounds, and sometimes a search round provides only duplicate documents, providing zero new information to seek the information. Our meta-statistics also show that the average number of new documents per round is only 2 to 3 (among 5) for most of the questions from BrowseComp-Plus, meaning that half of the retrieved documents are duplicates.

Intuitively, such a large portion of duplicate documents unnecessarily wastes the context capacity, but how much does it influence the effectiveness of information seeking in practice? We ablate its impact on Tongyi with BM25, Qwen3-Embedding-8B and Reason-ModernColBERT.[8]

Table 1 reports accuracy on BrowseComp-Plus in the original setting (w/o filter) versus removing duplicate documents from the retrievers' returned list (w/ filter), where we see the system benefit from removing duplicate documents with all evaluated retrievers. For example, with Qwen3-8B, it increases the overall accuracy from 0.5530 to 0.5988. If only looking at the questions that hit the context limit (thus with average accuracy to be 0), removing duplicate documents improves the accuracy to 0.2905, meaning that 29% of context limit could be resolved by saving LLM from duplicate contents.

[8]with get-document

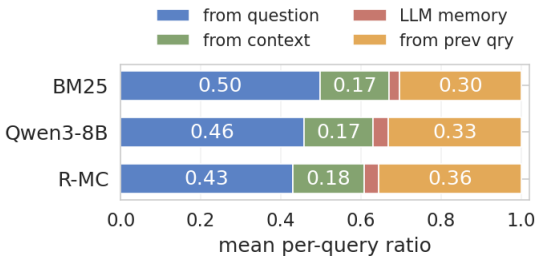

Figure 3: Aggregated query provenance ratios on BrowseComp-Plus while using Tongyi and three retrievers. Colors match to their corresponding category in the query provenance panel. "Qwen3-8B": Qwen3-Embedding-8B, "R-MC": Reason-ModernColBERT.

Limited Context Contribution in Query Formulation. Exact document duplication captures one form of redundancy in search trajectories; another is whether retrieved documents shape later query formulation. In HAWKEYE, this behavior is shown through the query provenance view in Figure 1 and the final meta-statistics panel, where we observe that most of the query tokens are traced to the original questions, then to previous queries, and finally to retrieved context.

Figure 5 shows an example of such attribution at dataset-level, yet, to help to characterize this behavior in paper, we further present an aggregated view in Figure 3. Across all retrievers, the original user question contributes to more than 40% of all query tokens, indicating that the agent's search behavior remains strongly anchored to the initial information need. By contrast, only around 20% of query tokens are newly introduced from retrieved context, suggesting that retrieved evidence is only sparsely converted into follow-up search queries. Meanwhile, around 30% of query tokens are reused from previous queries. Together, these patterns suggest that many later queries are reformulations of the agent's existing search vocabulary, rather than heavily relying on context-driven expansions from newly retrieved documents.

7 Related Work

Agentic Search. Agentic search extends retrieval-augmented generation from a single retrieve-then-answer step into a multi-turn process in which an LLM repeatedly issues queries, inspects retrieved evidence, and decides how to continue searching (Lewis et al., 2020). Early tool-using and browser-assisted systems such as WebGPT, Toolformer, ReAct, and Self-RAG showed that language models can benefit from external tools and retrieval during generation (Nakano et al., 2022; Schick et al., 2023; Yao et al., 2023; Asai et al., 2024). Recent deep-research agents push this idea toward harder information-seeking tasks, where success requires sustained exploration, multi-hop reasoning, and adaptive query reformulation across many retrieval turns (Wei et al., 2025; Zhou et al., 2025). This shift has motivated new benchmarks and infrastructure for evaluating long-horizon search behavior (Wei et al., 2025; Chen et al., 2025b; Coelho et al., 2025). In parallel, systems and training frameworks such as Search-R1 and WebSailor improve agents' ability to interleave reasoning with search tool use (Jin et al., 2025; Li et al., 2025). Our work builds on this line of research, but studies what happens inside the search trajectory: rather than only reporting final answer accuracy or aggregate tool-use counts, we provide tools to help every practitioner easily inspect how LLM-issued queries evolve, what information contributes to the query reformulation, and how retrieval behavior changes over the trajectory.

Analysis of Search Trajectories. Traditional search-log analysis has long studied query reformulation, session structure, and within-session learning in human web search (Eickhoff et al., 2014; Boldi et al., 2011; Huang and Efthimiadis, 2009). Agentic search introduces a related but distinct setting: the searcher is an autonomous LLM agent, and the observable trace consists of generated subqueries and retrieved evidence rather than clicks, dwell time, or explicit human feedback. Recent work on agentic search logs studies this behavior at scale. For example, Ning et al. (2026) analyze 14M+ real search requests from DeepResearch-Gym, label session intents and query reformulation types, and propose Context-driven Term Adoption Rate (CTAR) to measure whether newly introduced query terms are traceable to previously retrieved evidence, which we discussed in detail in Section 4.2 on the metric design difference. Additionally, HAWKEYE provides an interactive visual interface for inspecting individual questions and dataset-level patterns together instead of only producing corpus-level statistics. These visual diagnostics complement benchmark and log-analysis studies by helping practitioners identify trajectory-level behaviors on any customized agentic search runs, where we provide some examples in Section 6, such as repeated retrieval of already-seen documents and limited propagation of retrieved context into later queries.

8 Conclusion

HAWKEYE provides a visualization tool for opening the black box of agentic search trajectories by making intermediate queries, retrieved documents, and their relationships inspectable at both the per-question and dataset levels, which helps users understand search behavior beyond the final accuracy or the number of search calls. It exposes how an agent moves across topics, where query terms are derived from, and whether each retrieval round contributes new information.

Using HAWKEYE on BrowseComp-Plus, we identify two concrete trajectory-level patterns: repeated retrieval of already seen documents, which wastes context and can hurt accuracy, and that LLM-issued queries only sparsely stem from newly retrieved context. These findings suggest that improving agentic search requires not only stronger retrievers or more search calls, but also better mechanisms for avoiding redundant evidence and converting retrieved context into useful follow-up queries. We hope HAWKEYE can support more interpretable research on agentic search by easing the inspection process and helping a broader audience uncover trajectory-level behavior that aggregate metrics may obscure.

Limitations

LLM Coverage. Due to compute and budget constraints, our experiments use a single LLM backbone, `Tongyi-DeepResearch-30B-A3B`. We partially broaden the setting by evaluating diverse retrievers, including BM25, Qwen3-Embedding models of different sizes, and Reason-ModernColBERT. However, the trajectory-level patterns we observe may not generalize to agents built on other LLMs, prompting strategies, or search policies.

Lexical Attribution. The query provenance analysis relies on conservative lexical matching to attribute query tokens to the original question, previous queries, retrieved context, or model parametric memory. While this is an intentional design to make the attribution easy to inspect, it can miss semantic reuse when an agent paraphrases retrieved evidence or when the key information is reformed into other languages.

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 *The Twelfth International Conference on Learning Representations*.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. 2011. Query reformulation mining: models, patterns, and applications. *Inf. Retr.*, 14(3):257–289.

Ingwer Borg and Patrick J. F. Groenen. 1997. *Modern Multidimensional Scaling: Theory and Applications*. Springer Series in Statistics. Springer, New York.

Antoine Chaffin. 2025. Reason-moderncolbert.

Shan Chen, Pedro Moreira, Yuxin Xiao, Sam Schmidgall, Jeremy Warner, Hugo Aerts, Thomas Hartvigsen, Jack Gallifant, and Danielle S. Bitterman. 2025a. Medbrowsecomp: Benchmarking medical deep research and computer use. *Preprint*, arXiv:2505.14963.

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. 2025b. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *Preprint*, arXiv:2508.06600.

João Coelho, Jingjie Ning, Jingyuan He, Kangrui Mao, Abhijay Paladugu, Pranav Setlur, Jiahe Jin, Jamie Callan, João Magalhães, Bruno Martins, and Chenyan Xiong. 2025. Deepresearchgym: A free, transparent, and reproducible evaluation sandbox for deep research. *Preprint*, arXiv:2505.19253.

Carsten Eickhoff, Jaime Teevan, Ryen White, and Susan Dumais. 2014. Lessons from the journey: a query log analysis of within-session learning. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, page 223–232, New York, NY, USA. Association for Computing Machinery.

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.

Jeff Huang and Efthimis N. Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 77–86, New York, NY, USA. Association for Computing Machinery.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

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 *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Websailor: Navigating super-human reasoning for web agent. *Preprint*, arXiv:2507.02592.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

Jingjie Ning, João Coelho, Yibo Kong, Yunfan Long, Bruno Martins, João Magalhães, Jamie Callan, and Chenyan Xiong. 2026. Agentic search in the wild: Intents and trajectory dynamics from 14m+ real search requests.

Björn Ottosson. 2020. A perceptual color space for image processing. `https://bottosson.github.io/posts/oklab/`. Introduces Oklab and its cylindrical form, Oklch.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Qwen Team. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Tongyi DeepResearch Team. 2025b. Tongyi deepresearch: A new era of open-source ai researchers. https://github.com/Alibaba-NLP/DeepResearch.

W3C. 2024. CSS Color Module Level 4. https://www.w3.org/TR/css-color-4/. W3C Recommendation.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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 *International Conference on Learning Representations (ICLR)*.

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.

A Interplay between Query Provenance and Document Novelty

Here we present a dataset-level view of the interplay between query provenance and document novelty. As mentioned in Section 3, the three aspects visualized in the Question Panel of HAWKEYE may provide complementary information. For example, when *query provenance* shows that a query is largely derived from previous queries, *document novelty* helps assess whether such query reuse is productive for finding new information.

Figure 4 shows the ratio of query provenance from different sources versus number of new documents retrieved per round. The x-axis shows the ratio of query provenance from different sources, while the y-axis on the right shows the number of new documents retrieved per round and y-axis on the left shows which retriever is used in agentic search. The color of the bars matches the color of the query provenance in the Question Panel.

On one side, the figure shows that fewer novel retrieved documents per round correspond to higher query provenance from previous queries: when there are over 40% tokens from previous queries, the number of new documents retrieved per round drops to only 1 or even 0 per round (in a maximum of 5). On the other side, a small fraction of reused query tokens ($< 10\%$), together with question token and new tokens from context generally correspond to full utilization of retrieved documents.

B Demo of Detailed Question Page

Figure 5 shows a demo of the detailed question page in HAWKEYE. The top is the default display when clicked on a question panel (as shown in Figure 1), and the bottom highlights the information flow surrounding the query tokens when a query entry is hovered. Such information flow composed of two parts: (1) the highlighted blue tokens on the left question panel, which indicates query tokens from the user question; (2) the green and orange arcs on the right panel, connecting the query entries, which indicates the information flow in-between context and queries. Note that the arcs not only connect the query with its previous entries (i.e., the sources that contribute to itself), but also its future queries (the queries that itself contributes to). This allows users to find "sub-trajectories" that lurks inside the entire trajectory. With the demo page as an example, we observe that there are usually many independent paths within one query trajectory, which does not necessarily start from the first LLM-issued query or lead to the final round of search. For example, the demo figure shows a "dead-end" path, which contributes to no future queries even though there are many.

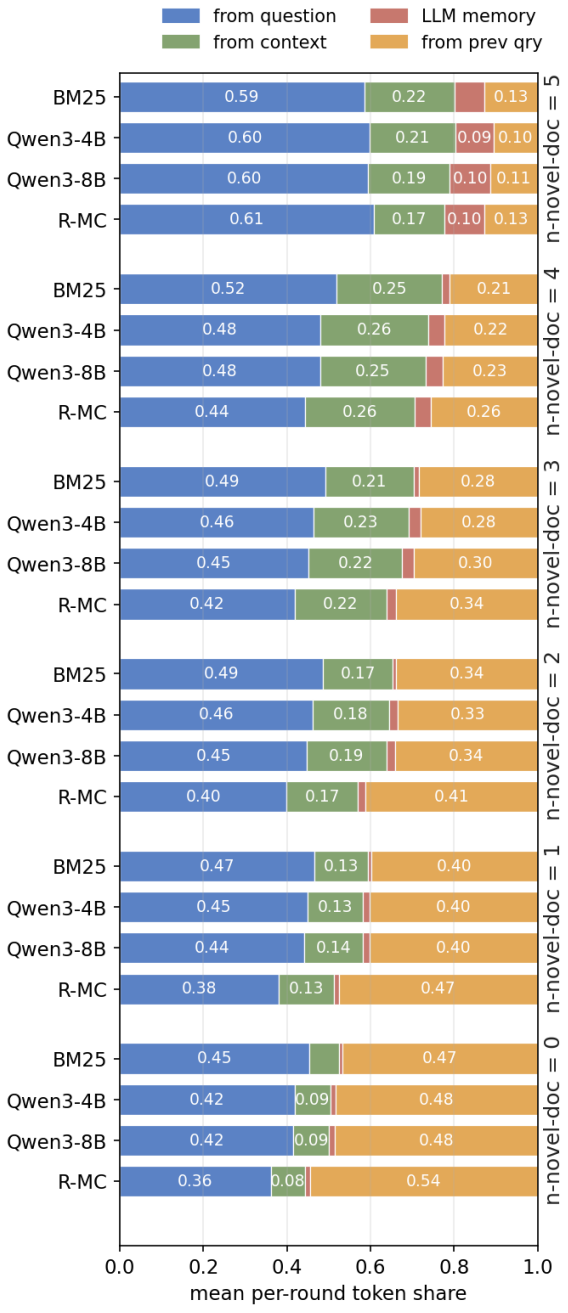

Figure 4: Ratio of query provenance from different sources (x-axis) versus number of new documents retrieved per round (labeled on y-axis on the right). "R-MC" stands for Reason-ModernColBERT, "Qwen3-" stands for Qwen3-Embedding- models. All experiments are conducted with Tongyi-30B as the LLM backbone.

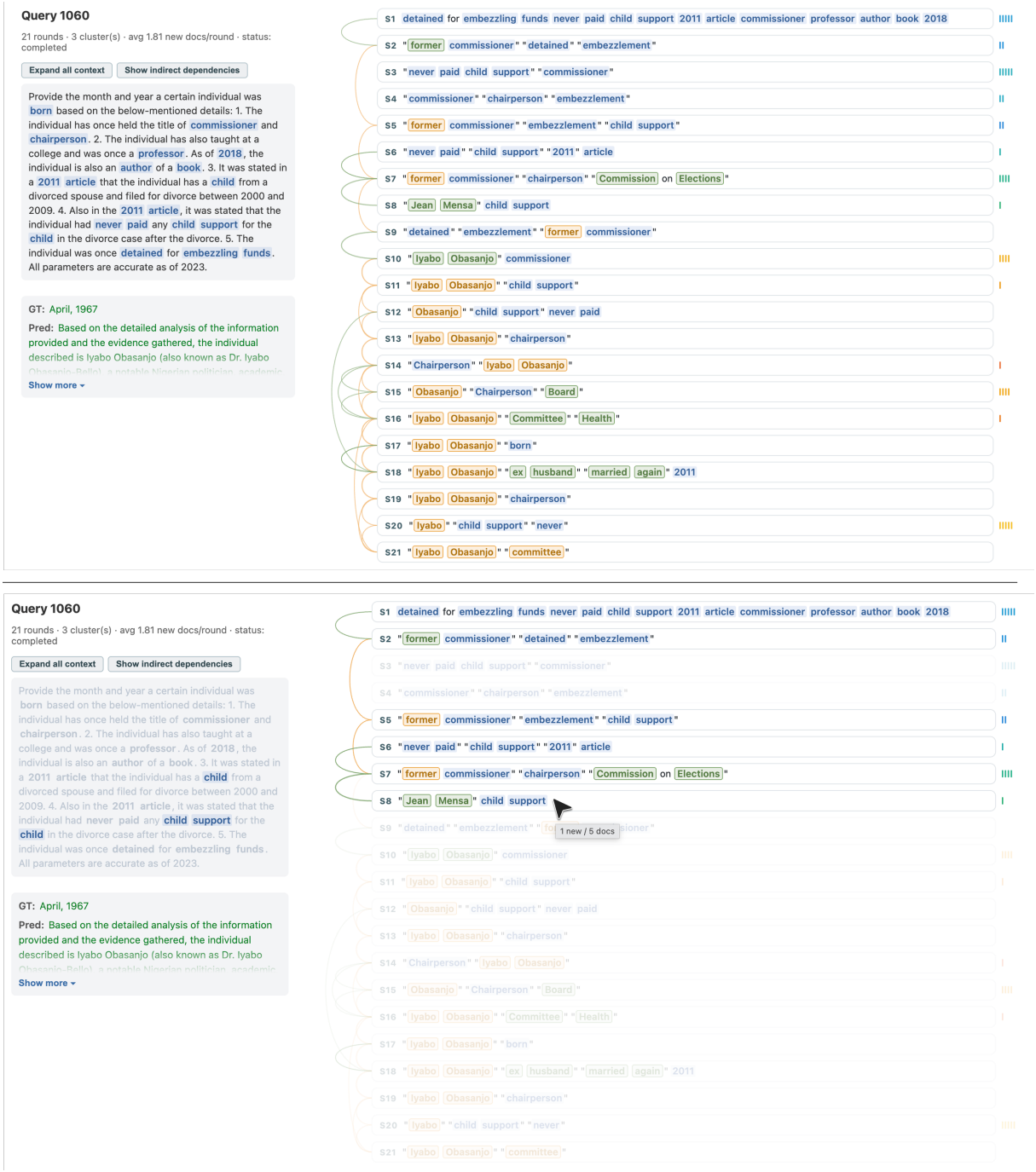

Figure 5: Demo of the detailed question page in HAWKEYE. Left: Default page display; Right: Highlighting the sources of the query tokens when a query entry is hovered.

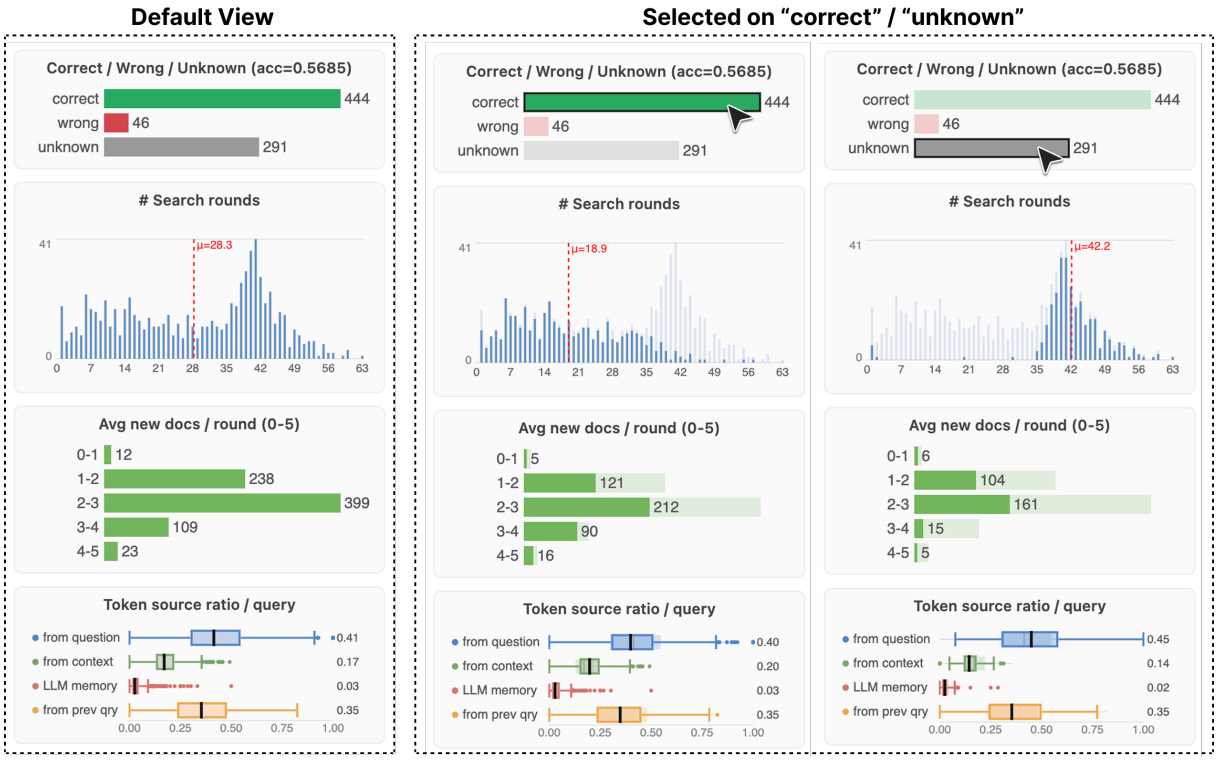

Figure 6: Demo of the detailed question page in HAWKEYE. Left: Default meta statistics panels; Middle and Right: After selecting the “correct” or “unknown” categories in the first panel, the meta statistics panels are updated to the statistics of questions under the selected category. (Note: The differences after selection in Panel 4 may be easier to see when zoomed in due to the panel’s rich use of color.)

C Demo of Meta Statistics Panels

As mentioned in Section 3, the first meta-statistics panel reports the accuracy breakdown of *correct*, *wrong*, and *unknown* evaluation result, along with the overall accuracy of the run. After selecting the “correct” or “unknown” categories in the first panel, not only the displayed Question Panels are filtered to only show the questions under the selected category, but also the other three meta statistics panels are updated to show the statistics of selected questions.

Figure 6 shows a demo on how the first meta-statistics panel can be used to filter the other three panels. The left panel is the default display, while the middle and right panels are the displays after selecting the “correct” or “unknown” categories in the first panel: which highlights statistics of questions under the selected category and moves the overall dataset-level statistics to the background (shown in less saturated colors). This provides a quick way to inspect the associated features when the question is correctly or incorrectly answered. For example, number of search rounds (panel 2) follows two distinctively different Gaussians for “correct” and “unknown” questions, where the “correct” questions tend to have much fewer search rounds. Similarly, the correctly answered questions tend to have more novel documents per round (panel 3) and higher query provenance from question and context (panel 4).