Revisiting Text Ranking in Deep Research

Chuan Meng
The University of Edinburgh
Edinburgh, United Kingdom
chuan.meng@ed.ac.uk

Litu Ou
The University of Edinburgh
Edinburgh, United Kingdom
litu.ou@ed.ac.uk

Sean MacAvaney
University of Glasgow
Glasgow, United Kingdom
sean.macavaney@glasgow.ac.uk

Jeff Dalton
The University of Edinburgh
Edinburgh, United Kingdom
jeff.dalton@ed.ac.uk

Abstract

Deep research has emerged as an important task that aims to address hard queries that need extensive open-web exploration. To tackle it, most prior work equips large language model (LLM)-based agents with opaque web search APIs, enabling agents to iteratively issue search queries, retrieve external evidence, and reason over it. Despite search's essential role in deep research, black-box web search APIs leave the behaviour of established text ranking methods in deep research largely unclear. To fill this gap, we reproduce key findings and best practices for text ranking methods in deep research. We examine their effectiveness from three perspectives: (i) retrieval units (documents vs. passages), (ii) pipeline configurations (different retrievers, re-rankers, and re-ranking depths), and (iii) query characteristics (the mismatch between agent-issued queries and the training queries of text rankers). We perform experiments on BrowseComp-Plus, a deep research dataset with a fixed corpus, evaluating 2 open-source agents, 5 retrievers, and 3 re-rankers. We find that agent-issued queries typically follow web-search-style syntax (e.g., quoted exact matches), favouring lexical, learned sparse, and multi-vector retrievers; passage-level units are more efficient under limited context windows, and avoid the difficulties of document length normalisation in lexical retrieval; re-ranking is highly effective. We further propose a query-to-question (Q2Q) method that translates agent-issued queries into natural-language questions, significantly reducing the query mismatch.

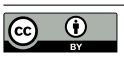
CCS Concepts

• **Information systems** → **Information retrieval**.

Keywords

Text ranking, Retrieval, Re-ranking, Deep research, Agentic search

ACM Reference Format:
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 (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3805712.3808557



SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3808557

1 Introduction

Text ranking is a core part of information retrieval (IR) [22, 28, 38, 42, 43, 58, 69]. It aims to produce an ordered list of texts retrieved from a corpus in response to a query [26]. Recently, the IR community has witnessed the emergence of *deep research* [46] scenarios. Deep research aims to answer multi-hop, reasoning-intensive queries that require extensive exploration of the open web and are difficult for both humans and large language models (LLMs) [55]. To tackle this task, a growing number of studies build agents that interact with live web search APIs to obtain information [23, 27, 70]. Such agents typically use LLMs as their decision-making core and perform multiple rounds of chain-of-thought (CoT) reasoning [56] and search invocations [18, 24, 47].

Motivation. Despite the essential role of search in deep research, how existing text ranking methods perform in this setting remains poorly understood. Most prior studies rely on black-box live web search APIs [61, 70], which lack transparency, thereby hindering systematic analysis and a clear understanding of the contribution of search components [6, 17].

Research goal. In this paper, we examine *to what extent established findings and best practices in text ranking methods are generalisable to the deep research setup*. We identify three research gaps in text ranking for deep research that have received limited attention.

First, *passage-level information units have received little attention in deep research*. Most existing studies build deep research agents that search and read at the document level (i.e., full web pages). Because feeding full web pages to LLM-based agents quickly exhausts the context window, prior work [6, 45] typically returns truncated documents to agents, which may remove relevant content and lead to information loss. Although prior work introduces an additional full-document read tool that agents can invoke to access complete documents [6], it adds system complexity. There is strong motivation to explore passage-level units in deep research: (i) their concise nature makes them more efficient under limited context-window budgets; (ii) they allow agents to access any relevant segments within a document, avoiding information loss from document truncation; (iii) passages can enhance lexical retrieval by avoiding the difficulties of document length normalisation [19]; and (iv) a large body of neural retrievers has been developed for passage retrieval [11, 12, 22, 43], but it remains unclear how well they perform in deep research. To address this gap, we ask **RQ1:** *To what extent are existing retrievers effective in deep research under*

passage-level and document-level retrieval units? In this RQ, we revisit key findings that neural retrievers often underperform lexical methods (e.g., BM25 [42]) on out-of-domain data [51], and that learned sparse and multi-vector dense (a.k.a. late-interaction) retrievers generalise better than single-vector dense retrievers on out-of-domain data [11, 51].

Second, *our understanding of re-ranking in deep research remains limited.* Re-ranking plays an important role in lifting relevant documents to the top of ranked lists in traditional search settings [37]. It remains unclear whether widely-used re-ranking methods [28, 38, 58] provide consistent benefits when the content consumer is an LLM-based agent rather than a human user. Despite recent work [45] examining one specific re-ranking method with a single retriever, a systematic evaluation across various ranking configurations is still lacking in deep research. To address this gap, we ask **RQ2:** *To what extent is a re-ranking stage effective in deep research under different initial retrievers, re-ranker types, and re-ranking cut-offs?* In this RQ, we revisit established findings that re-ranking effectively improves ranking performance [28, 37, 51], deeper re-ranking generally produces better results [30], and reasoning-based re-rankers outperform their non-reasoning counterparts [58, 64].

Third, *the potential mismatch between agent-issued queries and the queries used to train existing text ranking methods remains underexplored.* Many text ranking methods in the IR community are trained on natural-language-style questions, such as those in MS MARCO [4]. However, agent-issued queries may not align with the queries these methods expect. Prior studies show that a query-format mismatch between training and inference can significantly hurt neural retrieval quality [33, 71]. It remains unclear how such a potential mismatch affects the performance of existing text ranking methods in deep research. To address this gap, we ask **RQ3:** *To what extent does the mismatch between agent-issued queries and the training queries used for text ranking methods affect their performance?*

Experiments. We perform experiments on BrowseComp-Plus [6], a deep research dataset that provides a fixed document corpus and human-verified relevance judgments. To ensure broad coverage, we use widely-used retrievers spanning 4 main paradigms in modern IR: lexical-based sparse (BM25 [42]), learned sparse (SPLADE-v3 [22]), single-vector dense (RepLLaMA [28], Qwen3-Embed [69]), and multi-vector dense retrievers (ColBERTv2 [43]). For re-ranking, we select methods that trade off effectiveness and efficiency at 3 operational points: a relatively inexpensive re-ranker (monoT5-3B [38]), an LLM-based re-ranker (RankLLaMA-7B [28]), and a CoT-based reasoning re-ranker (Rank1-7B [58]), which generates additional reasoning tokens. We use two open-source LLM-based agents: gpt-oss-20b [1] and GLM-4.7-Flash (30B) [67].

Findings. For RQ1, our findings are fivefold: (i) The concise nature of passage-level units enables more search and reasoning iterations before reaching context-window limits, resulting in higher answer accuracy than document-level units (without a full-document reader), particularly for the gpt-oss-20b agent that has shorter context windows. (ii) BM25 on the passage corpus outperforms neural retrievers in most cases (gpt-oss-20b with BM25 achieves the highest accuracy of 0.572 across all retrieval settings in our study). We find agent-issued queries tend to follow a *web-search style with keywords, phrases, and quotation marks for exact matching* (see Table 5), favouring lexical retrievers such as BM25. (iii) On the document corpus, BM25 performs worst under the parameter settings used in prior work [6]. We find that these parameters lack proper document-length normalisation. With document-oriented parameters, however, BM25 becomes highly competitive. This highlights BM25's sensitivity to length normalisation and the advantage of passage-level units, which reduce reliance on document-length normalisation. (iv) learned sparse [22] and multi-vector dense [43] retrievers with only millions of parameters generalise better to web-search-style queries than 7B/8B single-vector dense models. (v) Enabling a full-document reader on truncated documents generally improves answer accuracy and reduces search calls, suggesting that the reader complements truncated inputs. In contrast, adding the reader to the passage corpus slightly degrades performance, likely because passage retrieval already provides access to any segments within a document, rendering the reader redundant.

For RQ2, re-ranking consistently improves ranking effectiveness and answer accuracy while reducing search calls, confirming its important role in deep research. These gains are further amplified by deeper re-ranking depths. Notably, the BM25–monoT5-3B pipeline with gpt-oss-20b achieves the best results in our work, reaching 0.716 recall and 0.689 accuracy. Despite using only a 20B agent with BM25 and a 3B re-ranker, this setup approaches the 0.701 accuracy of a GPT-5–based agent (Table 1 in [6]). The reasoning-based Rank1 [58] shows no clear advantage over non-reasoning methods, as it often misinterprets the intent of keyword-rich web-search queries, limiting the benefits of reasoning.

For RQ3, we propose a *query-to-question* (Q2Q) method to translate agent-issued web search queries into natural-language questions (similar to MS MARCO-style questions [4]), significantly improving neural retrieval and re-ranking performance. This indicates that the mismatch between agent-issued queries and queries used for training neural rankers can severely degrade neural ranking effectiveness. Mitigating this training–inference query mismatch is therefore critical for improving neural rankers in deep research.

Contributions. Our main contributions are as follows:

- To the best of our knowledge, we are the first to reproduce a comprehensive set of text ranking methods in deep research.
- We construct a passage corpus for the recent deep research dataset BrowseComp-Plus.
- Experiments across 2 open-source agents, 5 retrievers, and 3 re-rankers reveal the effectiveness of the following components in deep research: passage-level information units, retrievers suited to web-search-style queries, re-ranking, and mitigation of the training–inference query mismatch in neural ranking.
- We open-source our code, data, and all agent-generated traces at https://github.com/ChuanMeng/text-ranking-in-deep-research.

2 Task definition

Text ranking has been extensively studied in ad-hoc search, which typically follows a single-shot paradigm over user-issued queries. Given a query q_u issued by a user and a corpus of documents $C = \{d_1, d_2, \ldots, d_n\}$ with n documents, the goal of a text ranking method f is to return a ranked list $D \subseteq C$ of size k, i.e., $D = f(q_u, C)$.

Text ranking in deep research. We follow the ReAct [66] paradigm, widely used in recent work [6, 60], to define text ranking in

deep research. Given a query q_u issued by a user, a deep research agent A takes q_u as input and performs multiple iterations of reasoning and search before producing the final answer a. At iteration t, the agent generates a reasoning trace s_t that determines whether to output the final answer a or invoke search to gather information. If the agent decides to invoke search, it issues a search query q_t to a text ranking method f, which returns a ranked list D_t:

$$\begin{aligned} q_t &= A(q_u, s_0, q_0, D_0, \ldots, s_t), \\ D_t &= f(q_t), \end{aligned} \tag{1}$$

where s_0, q_0 are the reasoning trace and query generated by the agent at the initial iteration, respectively; D_0 is the ranked list returned by f in response to q_0. The ranked list D_t with k documents is then fed back to the agent to produce the reasoning trace at the next iteration $t+1$:

$$s_{t+1} = A(q_u, s_0, q_0, D_0, \ldots, s_t, q_t, D_t). \tag{2}$$

Note that some agents may perform multiple consecutive search invocations or reasoning steps; for simplicity, the above definition assumes alternating reasoning and search steps.

3 Methodology

This section presents our research questions, experimental design, and experimental setup, including the agents and ranking methods, dataset, evaluation protocol, and our proposed method.

3.1 Research questions and experimental design

RQ1 To what extent are existing retrievers effective in deep research under passage-level and document-level retrieval units?

To address **RQ1**, we reproduce widely-used retrievers from different categories (see Section 3.2.2) in deep research, and compare their performance on passage and document corpora. BrowseComp-Plus [6], the dataset used in this study, provides only a document corpus; we construct a passage corpus (see Section 3.2.4). When using document-level retrieval units, prior work [6, 45] typically feeds truncated retrieved documents to agents to avoid exhausting the context window; Chen et al. [6] further introduces a full-document reader tool that allows agents to access complete documents when needed. Accordingly, we evaluate three settings: (i) agents retrieve documents but read truncated versions; (ii) same as (i), but with a full-document reader tool; and (iii) agents retrieve and read passages. In addition, we consider a setting in which agents retrieve and read passages, and can choose to invoke the full-document reader to read the source document of a retrieved passage.

RQ2 To what extent is a re-ranking stage effective in deep research under different initial retrievers, re-ranker types, and re-ranking cut-offs?

To address **RQ2**, we reproduce widely-used re-rankers (see Section 3.2.2), and evaluate text ranking pipelines under various configurations, including different initial retrievers, different re-rankers, and different re-ranking depths.

RQ3 To what extent does the mismatch between agent-issued queries and the training queries used for text ranking methods affect their performance?

To address **RQ3**, we compare the performance of text ranking methods using agent-issued search queries with their performance using natural-language questions similar to those in MS MARCO [4], on which many neural rankers are trained. We propose a query-to-question (Q2Q) method, which translates agent-issued web search queries into natural-language questions; see Section 3.2.5 for details.

3.2 Experimental setup

3.2.1 Deep research agents. We use two LLMs as deep research agents with model sizes that are feasible to a broad range of research groups. Specifically, we follow [6] to use gpt-oss-20b [1], and use the recently released GLM-4.7-Flash [67]. Both LLMs have been trained to invoke web search, which is crucial for deep research.

- **gpt-oss-20b** [1] is an OpenAI's open-weight LLM. It is pre-trained and subsequently post-trained for reasoning (i.e., using chain-of-thought) and tool use. For tool use, the model is trained to interact with the web via search and web page opening actions. It supports a maximum context window and output length of 131,072 tokens.
- **GLM-4.7-Flash (30B)** [67] is an open-source LLM developed by Z.ai. It has a pre-training stage, followed by a mid-training phase (improves coding & reasoning) and a post-training phase. During post-training, the model is explicitly trained for web search via reinforcement learning (RL). It supports a context window of 202,752 tokens and a maximum output length of 128,000 tokens.

Note that our work focuses on reproducing text ranking methods in the deep research scenario; considering other LLMs as deep research agents or scaling sizes is beyond the scope of this paper.

3.2.2 Text ranking methods to be reproduced. We employ a set of widely-used and representative text-ranking methods in the IR community, including retrievers and re-rankers.

Retrievers. We use widely-used retrievers spanning 4 main paradigms in modern IR: lexical-based sparse, learned sparse, single-vector dense, and multi-vector dense retrievers:

- **BM25** [42] is a lexical retriever based on vocabulary-level vectors using the bag-of-words approach for queries and documents.
- **SPLADE-v3** [22] is a learned sparse retriever that trains BERT [10] to predict sparse vectors over BERT's vocabulary list for queries and documents.
- **RepLLaMA** [28] is a single-vector dense retriever; it fine-tunes Llama 2 [52] using LoRA [16] to produce a single embedding vector for each query and document. It appends an end-of-sequence token to each query or document input and uses the hidden state of the last model layer corresponding to this token as the embedding. Its embedding dimension is 4,096.
- **Qwen3-Embed-8B** [69] is a single-vector dense retriever. It belongs to the Qwen3-Embedding series (0.6B, 4B, and 8B), and we use the largest variant. The series features fine-tuning of Qwen3 LLMs [62] on synthetic training data across multiple domains and languages, generated by the Qwen3 models themselves. It follows RepLLaMA to obtain query and document embeddings and shares the same embedding dimension.
- **ColBERTv2** [43] is a multi-vector retriever. It trains BERT [10] to produce embeddings for each token in the query and the document, and models relevance as the sum of the maximum similarities between each query vector and all document vectors.

Note that the training data and input length configuration of Qwen3-Embed-8B differ substantially from those of other neural retrievers (SPLADE-v3, ColBERTv2, and RepLLaMA): (i) Qwen3-Embed-8B is

Table 1: Statistics of the BrowseComp-Plus dataset [6]. Length is measured in tokens using the Qwen3 [62] tokeniser.

# Q	Avg. Q Len.	Corpus Type	# Items	Avg. Item Len.
830	132.19	Document	100,195	7,845.55
		Passage	2,772,255	279.64

trained on fully synthetic data generated by Qwen3, whereas the others share the same training dataset, namely the training data of the MS MARCO V1 passage ranking corpus [4]; (ii) Qwen3-Embed-8B is trained to support document lengths of up to 32,000 tokens, whereas the other neural ones are trained for passage-level inputs.

Re-rankers. We select methods representing three effectiveness–efficiency trade-offs: a relatively inexpensive model (monoT5-3B [38]), an LLM-based re-ranker (RankLLaMA-7B [28]), and a CoT-based reasoning re-ranker (Rank1-7B [58]), which requires reasoning token generation. All of them are pointwise re-rankers, which independently assign a relevance score given a query and a document:

- **monoT5-3B** [38] is a non-reasoning-based re-ranker that fine-tunes T5 [41] to output either "true" or "false" to indicate relevance. The probability assigned to the "true" token is used as the relevance score. monoT5 is available in base (220M), large (770M), and 3B variants; we use the 3B model.
- **RankLLaMA-7B** [28] is a non-reasoning-based re-ranker that fine-tunes Llama 2 [52] with LoRA [16] to project the representation of the end-of-sequence token to a relevance score.
- **Rank1-7B** [58] is a reasoning-based re-ranker that fine-tunes Qwen 2.5 [63] to generate a reasoning trace before producing a "true"/"false" decision ("<think> ... </think> true/false"). The ground truth training reasoning traces are generated by DeepSeek-R1 [13] on MS MARCO [4]. As in monoT5, the probability assigned to the "true" token is used as the relevance score.

All re-rankers are trained on the MS MARCO V1 passage ranking dataset [4] and trained to operate on passage-level inputs. Note that evaluating larger re-ranker variants (e.g., RankLLaMA-13B and Rank1-14B/32B) is beyond the scope of this work.

3.2.3 Dataset. We use BrowseComp-Plus [6], a deep research dataset built on BrowseComp [55], which comprises 1,266 fact-seeking, reasoning-intensive, long-form queries together with their answers; the answers are generally short and objective, making answer evaluation easier. BrowseComp-Plus adds a document corpus and human-verified relevance judgments via a three-stage process. (i) each question–answer pair is sent to OpenAI o3 to identify clues (a clue is a part of the question) useful for deriving the answer and to return supporting web documents; (ii) human annotators verify whether each clue is well supported by its supporting documents and whether the combination of clues and documents enables answering the question; annotators revise the clues and supporting documents when necessary; (iii) the final document corpus is constructed by including all human-verified supporting documents, together with mined hard-negative documents. Some queries are removed in (i) and (ii) when OpenAI o3 fails to return valid clues/supporting documents, or it is too hard for human annotators to correct them, resulting in a final set of 830 queries. The dataset provides two types of relevance judgments: *gold* and *evidence*. Gold documents contain the final answers (not necessarily exact matches) to the queries, while evidence documents include the gold documents and documents supporting intermediate reasoning steps. On average, each query has 2.9 gold, 6.1 evidence, and 76.28 negative documents. Table 1 reports detailed statistics.

3.2.4 Passage corpus construction. To segment the documents into passages, we follow [9, 40] to split each document in the original document corpus of BrowseComp-Plus into canonical passages of at most 250 words using the spaCy toolkit with the "en_core_web_sm" model; we use the publicly available code.[1] The statistics of the passage corpus are shown in Table 1. Each passage is assigned a new passage ID, and we record the mapping from each passage to its original document. Following [8], when a document title is available, we extract it and prepend it to the beginning of each corresponding passage to provide additional context.

3.2.5 Query-to-question (Q2Q) reformulation. To translate a web search query q_t issued by an agent at iteration t into a natural-language question, we define a query-to-question (Q2Q) reformulator g that takes q_t as input and outputs a natural-language question $\tilde{q}_t$, i.e., $\tilde{q}_t = g(q_t)$; then, $\tilde{q}_t$ is sent to a text ranking method f to return a ranked list $D_t = f(\tilde{q}_t)$. However, a web search query q_t may be ambiguous and may not clearly reflect the agent's search intent; relying solely on the query may therefore cause the reformulation to deviate from the agent's search intent. To provide additional context about the agent's search intent, we introduce another variant of Q2Q that includes the recent reasoning trace s_t generated by the agent, namely $\tilde{q}_t = g(q_t, s_t)$.

3.2.6 Evaluation. We follow the original BrowseComp-Plus evaluation protocol [6] and report *the number of search calls*, *recall*, and *accuracy*; we use the evaluation code released by the dataset authors.[2] Search calls denote the average number of search invocations per query. Recall measures the proportion of evidence documents returned across all search calls for a query. Accuracy is computed using an LLM-as-judge that compares an agent's final answer with the ground-truth answer. To analyse token-budget efficiency, we additionally report *completion rate*, i.e., the percentage of queries for which the agent outputs a final answer before reaching the maximum context-window or output-token limits; queries reaching these limits receive accuracy 0.

Evaluating on passage corpus. Because BrowseComp-Plus provides relevance judgments at the document level, they cannot be directly applied to the passage corpus. When evaluating passage retrieval, we therefore adopt the *Max-P* strategy [8], mapping retrieved passages to documents by assigning each document the maximum score amongst its retrieved passages.

3.2.7 Implementation details. Regarding the agent setup, for both gpt-oss-20b[3] and GLM-4.7-Flash (30B)[4], following [6], we set the maximum output length to 40,000 tokens and the maximum number of iterations to 100. We run gpt-oss-20b in the high reasoning-effort mode (the default setting); the reasoning effort of GLM-4.7-Flash (30B) is not configurable. Both agents are deployed locally using vLLM version 0.15, which satisfies the minimum version requirement of GLM-4.7-Flash.

[1] https://github.com/grill-lab/trec-cast-tools/tree/master/corpus_processing
[2] https://github.com/texttron/BrowseComp-Plus
[3] https://huggingface.co/openai/gpt-oss-20b
[4] https://huggingface.co/zai-org/GLM-4.7-Flash

Table 2: Sanity-check comparison of agent performance for gpt-oss-20b using Qwen3-Embed-8B as a search tool on the original BrowseComp-Plus document corpus. Results from two recent studies [6, 45] are shown alongside our replicated results. We also report results for GPT-5.2 (high reasoning mode), which tends to generate longer reasoning traces and search much more aggressively, causing 354 of 830 queries to reach the maximum iteration limit (100); these queries are assigned an accuracy of 0, resulting in lower overall accuracy.

Agent	Search calls	Recall	Acc.	vLLM
gpt-oss-20b [6]	23.87	0.493	0.346	v0.9.0.1
gpt-oss-20b [45]	29.63	0.557	0.422	v0.13
gpt-oss-20b (ours)	30.14	0.570	0.421	v0.15
GPT-5.2 (ours)	73.83	0.747	0.451	-

Regarding the text ranking setup, we follow prior work [6, 45]: after each search call, we return the top-5 ranked documents (or passages) to the agent and truncate each returned document to its first 512 tokens; this prevents long documents from exhausting the context window: with an average document length of 8K tokens (Table 1), a 130K-token context window (gpt-oss-20b) could hold only about 16 full documents. We use Pyserini's BM25 with its default parameters ($k_1 = 0.9$, $b = 0.4$), following [6]. We implement SPLADE-v3 (naver/splade-v3), RepLLaMA (castorini/repllama-v1-7b-lora-passage)[5], RankLLaMA-7B (castorini/rankllama-v1-7b-lora-passage), and Qwen3-Embed-8B (Qwen/Qwen3-Embedding-8B) using Tevatron.[6] We implement ColBERTv2 (colbert-ir/colbertv2.0) using PyLate [5].[7] We implement monoT5-3B (monot5-3b-msmarco) following PyTerrier_t5.[8] Rank1-7B (jhu-clsp/rank1-7b) is implemented following the original authors' implementation.[9] For document indexing, we follow [6] to set the maximum input length of Qwen3-Embed-8B to 4,096 tokens; the remaining neural retrievers use a maximum length of 512 tokens, as they are trained on passage-level inputs. For passage indexing, all retrievers use a maximum length of 512 tokens. We implement the query-to-question (Q2Q) reformulator (see Section 3.2.5) using gpt-oss-20b in the low reasoning-effort mode; we randomly sample natural language questions from TREC-DL 2019 [7] and include them in the prompt as examples to specify the desired question-style output. Experiments are conducted using NVIDIA RTX 6000 Ada (48 GB), H100 (80 GB), and H200 (141 GB) GPUs, subject to hardware availability.

4 Results and Discussions

4.1 Sanity check for reproduction

As a sanity check, we attempt to replicate the results reported in recent studies [6, 45]. We evaluate gpt-oss-20b (high-reasoning mode) with Qwen3-Embed-8B as the retriever on the original BrowseComp-Plus document corpus. The results are shown in Table 2. Our results obtained with the latest vLLM version (v0.15) are very similar to those produced using v0.13 in [45]. We observe that newer vLLM versions (v0.15 and v0.13) lead to higher performance compared to the old one (v0.9.0.1). Besides replicating gpt-oss-20b, we also test GPT-5.2, a current state-of-the-art commercial LLM. We find that GPT-5.2 requires substantially more iterations; 354 queries (830 queries in total) require over 100 iterations. Such a large number of iterations would incur significant API costs, making it less feasible for broader researchers and practitioners (all queries cost roughly $1,000 to $2,000); therefore, we do not use it in this work.

4.2 Retrievers on passage and document corpora

To answer **RQ1**, we compare two agents, gpt-oss-20b and GLM-4.7-Flash (30B), across different retrievers on both the passage and document corpora. For experiments on the document corpus, following [6, 45], we truncate each retrieved document to the first 512 tokens, to avoid exhausting the input context window. To mitigate truncation-induced information loss, we follow [6] to evaluate a setting where agents can access a full-document reader tool on the document corpus. We present the results in Tables 3 (gpt-oss-20b) and 4 (GLM-4.7-Flash). We have four main observations.

First, both agents achieve higher answer accuracy on the passage corpus than on the document corpus (without the full-document reader) across all retrievers, except Qwen3-Embed-8B, likely because it is trained on substantially longer documents (up to 32K tokens) and is less effective on passages due to distribution shift. Notably, the relative improvement on passages is more pronounced for the shorter-context gpt-oss-20b (131K context window) than for GLM-4.7-Flash (200K). E.g., with SPLADE-v3, gpt-oss-20b achieves 0.516 accuracy on passages, an 8.4% relative improvement over documents (0.476), whereas GLM-4.7-Flash gains 4.02%. We further observe that both agents issue more search calls on passages than on documents (e.g., 32.75 vs. 28.95 with gpt-oss-20b using SPLADE-v3). Moreover, for gpt-oss-20b, the completion rate is markedly higher on passages (0.980 vs. 0.824), while GLM-4.7-Flash shows similar completion rates across corpora; note that both models share the same output-token limit but differ in context-window limit. Taken together, these results suggest that passage-level units enable more search and reasoning iterations before reaching context-window limits, which in turn improves answer accuracy, particularly for agents with smaller context windows.

Second, when paired with the lexical retriever BM25, both agents achieve highly competitive performance on the passage corpus compared with neural rankers. For instance, gpt-oss-20b with BM25 attains the highest recall (0.616) and answer accuracy (0.572) on the passage corpus. The 0.572 accuracy is also the highest across all retrieval settings in Tables 3 and 4. Table 5 presents examples of agent-issued queries; they follow a web-search style, characterised by keywords, phrases, and quotation marks for exact matching, making them well suited to lexical retrievers such as BM25. In contrast, BM25 performs worst on the document corpus; we analyse this issue in more detail later in this section.

Third, single-vector dense retrievers (RepLLaMA, Qwen3-Embed-8B), despite their substantially larger model sizes, consistently underperform smaller BERT-based [10] learned-sparse (SPLADE-v3) and multi-vector dense (ColBERTv2) retrievers. This finding is consistent with prior work showing that SPLADE and ColBERT perform well across diverse query formats on the BEIR dataset [11, 51] that include keyword-rich queries or those requiring exact matching. ColBERT, in particular, has been shown to exhibit strong preferences for exact matching [36]. In contrast, single-vector approaches

[5] We also evaluated the RepLLaMA checkpoint trained on the MS MARCO document corpus (castorini/repllama-v1-7b-lora-doc); however, it performed worse on BrowseComp-Plus than the passage-trained checkpoint (repllama-v1-7b-lora-passage).
[6] https://github.com/texttron/tevatron
[7] https://github.com/lightonai/pylate
[8] https://github.com/terrierteam/pyterrier_t5
[9] https://github.com/orionw/rank1

Table 3: Agent performance across retrievers on BrowseComp-Plus. The agent is based on gpt-oss-20b; it supports a maximum context window of 131,072 tokens. "#Search" and "#GetDoc" denote the number of search and full-document reader calls, respectively. "Compl." denotes the completion rate, i.e., the percentage of queries for which the agent outputs a final answer before reaching the maximum context-window or output-token limits; queries reaching these limits receive accuracy 0. The best values in recall and accuracy are boldfaced, and the second-best ones are underlined.

Retriever	Passage corpus				Document corpus				Document corpus+GetDoc				
	#Search	Recall	Acc.	Comp.	#Search	Recall	Acc.	Comp.	#Search	#GetDoc	Recall	Acc.	Comp.
BM25	30.97	**0.616**	**0.572**	0.968	32.22	0.366	0.259	0.805	28.14	1.31	0.343	0.301	0.746
SPLADE-v3	32.75	0.545	0.516	0.980	28.95	0.628	0.476	0.824	24.64	1.65	0.602	0.529	0.829
RepLLaMA	36.13	0.449	0.406	0.961	30.69	0.514	0.363	0.786	26.86	1.46	0.476	0.399	0.816
Qwen3-Embed-8B	37.83	0.470	0.417	0.963	30.14	0.570	0.421	0.821	26.56	1.63	0.559	0.455	0.823
ColBERTv2	32.88	0.552	0.521	0.978	27.13	**0.633**	**0.481**	0.855	24.31	1.75	**0.612**	**0.538**	0.835

Table 4: Agent performance across different retrievers on BrowseComp-Plus. The agent is based on GLM-4.7-Flash (30B); it supports a maximum context window of 200,000 tokens. Metric definitions and formatting follow Table 3.

Retriever	Passage corpus				Document corpus				Document corpus+GetDoc				
	#Search	Recall	Acc.	Comp.	#Search	Recall	Acc.	Comp.	#Search	#GetDoc	Recall	Acc.	Comp.
BM25	34.73	**0.581**	0.445	0.863	26.83	0.309	0.196	0.923	21.86	3.00	0.282	0.263	0.947
SPLADE-v3	36.36	0.578	**0.466**	0.883	25.62	0.639	**0.448**	0.881	18.43	5.33	**0.597**	0.525	0.939
RepLLaMA	37.60	0.456	0.331	0.905	25.77	0.493	0.330	0.907	20.93	4.13	0.471	0.407	0.955
Qwen3-Embed-8B	37.47	0.482	0.357	0.886	26.25	0.580	0.374	0.882	20.12	4.34	0.482	0.456	0.956
ColBERTv2	36.41	0.571	0.464	0.882	26.06	**0.640**	0.430	0.878	18.85	5.30	0.595	**0.535**	0.955

Table 5: Examples of three search queries issued by the gpt-oss-20b and GLM-4.7-Flash agents for Query 781 on BrowseComp-Plus.

Agent	Search query
gpt-oss-20b	"90+7" attendance 61700 "Man United" "4-1" "90+4" "assist" "4–1" "Premier League" "2020"
GLM-4.7-Flash	"90'+4" football match attendance "61,888" football match Stockholm Vienna Prague "goal in the 6th minute" football

Table 6: Performance of the gpt-oss-20b agent across different retrievers on BrowseComp-Plus. Metric definitions and formatting follow Table 3.

Retriever	Passage corpus+GetDoc				
	#Search	#GetDoc	Recall	Acc.	Comp.
BM25	26.87	2.24	**0.556**	**0.542**	0.877
SPLADE-v3	27.54	2.11	0.532	0.510	0.895
RepLLaMA	31.82	1.77	0.399	0.369	0.868
Qwen3-Embed-8B	32.11	1.98	0.438	0.404	0.853

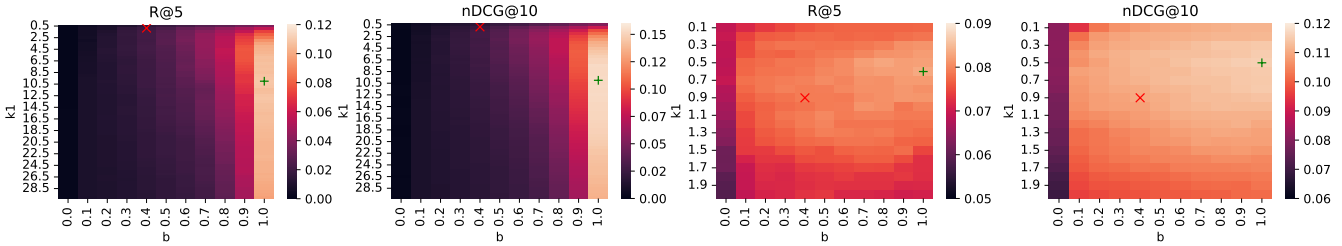
often struggle to adapt to new tasks and domains [51] and are theoretically constrained in representational capacity [57].

Fourth, enabling the full-document reader on the document corpus reduces search calls and recall but improves answer accuracy. E.g., with gpt-oss-20b using SPLADE-v3, accuracy increases from 0.476 to 0.529. This suggests that the reader compensates for information loss caused by document truncation. With the reader enabled on documents, gpt-oss-20b achieves performance comparable to the passage setting, while GLM-4.7-Flash generally surpasses the passage setting and maintains high completion rates, likely due to its longer context window, which allows processing of more complete documents. We also evaluate enabling the reader on the passage corpus using gpt-oss-20b with several retrievers; see results in Table 6. We find the reader slightly degrades performance (e.g., with BM25, accuracy decreases from 0.572 to 0.542), likely because passage-level units already provide direct access to relevant segments within a document, rendering the reader redundant.

Why does BM25 perform poorly on the document corpus? Prior work [19] shows that lexical document retrieval is sensitive to document-length normalisation. BM25 has two parameters k_1 and b [42]: k_1 controls term-frequency saturation (larger k_1 values lead to slower term frequency saturation, giving repeated terms more weight), while b controls document length normalisation (larger b values increase the penalty on longer documents). To investigate this issue, we evaluate the gpt-oss-20b agent using BM25 under two settings: (i) truncating each document to its first 512 tokens before indexing, thereby reducing the impact of document-length variation; and (ii) instead of using the default parameter ($k_1 = 0.9$, $b = 0.4$) following [6], we use document-retrieval-oriented BM25 parameters ($k_1 = 3.8$, $b = 0.87$), which increase length normalisation and have been shown to be effective for the MS MARCO document retrieval task[10]. We present the results in Table 7. Indexing only the first 512 tokens of each document substantially improves performance, yielding a 64.2% relative gain in recall and a 98.1% gain in answer accuracy, suggesting that reducing the effect of document-length normalisation benefits BM25 on documents. Using the document-oriented BM25 parameters also improves performance, with a 76.8% gain in recall and a 71.0% gain in accuracy.

To better understand the optimal BM25 parameter settings, we perform a grid search of retrieval performance on BrowseComp-Plus using the original full queries across different parameter values.

[10] https://github.com/castorini/anserini/blob/master/docs/experiments-msmarco-doc.md



(a) Document corpus (Recall@5) (b) Document corpus (nDCG@10) (c) Passage corpus (Recall@5) (d) Passage corpus (nDCG@10)

Figure 1: Heatmap from a grid search on BrowseComp-Plus using the original full queries (not end-to-end), showing the effectiveness (evaluated by evidence judgments) of BM25 under different hyperparameter settings. The red × denotes the default parameter setting following [6], while the green + denotes the best parameter setting found by the grid search. The lighter the colour, the higher the retrieval performance.

Table 7: Performance of the gpt-oss-20b agent using BM25 under different hyperparameter settings on BrowseComp-Plus. Index len. indicates which portion of each document is indexed, i.e., the full document or the first 512 tokens. $k_1 = 0.9$ and $b = 0.4$ are the default BM25 parameters, following [6]. The best values in recall and accuracy are boldfaced, and the second-best ones are underlined.

BM25 setting	Document corpus				
	Index len.	#Search	Recall	Acc.	Comp.
$k_1 = 0.9, b = 0.4$	full doc	32.22	0.366	0.259	0.805
$k_1 = 0.9, b = 0.4$	512 token	28.00	0.642	**0.513**	0.812
$k_1 = 3.8, b = 0.87$	full doc	28.66	0.601	0.443	0.839
$k_1 = 10, b = 1$	full doc	28.73	**0.647**	0.506	0.848

The results are shown in Figure 1. We find that larger values of b generally improve performance, i.e., penalising long documents with higher term frequencies is beneficial. Moreover, the performance gap between the default setting ($k_1 = 0.9$, $b = 0.4$) and the optimal parameter settings is substantially larger on the document corpus than on the passage corpus. E.g., on the document corpus, $k_1 = 10$ and $b = 1$ appear to be a sweet spot (see Figures 1a and 1b), and performance using it is substantially different from the default setting. We further evaluate gpt-oss-20b using BM25 with $k_1 = 10$ and $b = 1$. Table 7 shows that this configuration achieves the highest recall (0.647) across all settings in this section.

We note that k_1 and b are generally regarded as corpus-dependent parameters rather than query-specific ones [14]. Indeed, Figure 1 shows that strong length normalisation ($b = 1$) and large k_1 values are important for effective BM25 retrieval over BrowseComp-Plus documents. Precise tuning of these parameters is not required for strong effectiveness; in particular, a wide range of k_1 values performs well. Nonetheless, we urge readers to exercise some caution when comparing the tuned BM25 results with other retrievers, since the parameter selection was performed directly on all BrowseComp-Plus queries (due to the absence of a validation set).

4.3 Re-ranking in deep research

To answer **RQ2**, we evaluate two agents (gpt-oss-20b and GLM-4.7-Flash) that access ranking pipelines with varying retrievers, re-rankers, and re-ranking depths. Given the advantages of the passage corpus shown in Section 4.2, we perform all experiments on the passage corpus. We use BM25, SPLADE-v3, and Qwen3-Embed-8B as initial retrievers; monoT5-3B, RankLLaMA-7B, and Rank1-7B as re-rankers; and, following [45], we use three re-ranking depths (10, 20, and 50). Due to limited GPU resources, for GLM-4.7-Flash, we exclude Qwen3-Embed-8B, Rank1 and depth 20. We show the results for gpt-oss-20b and GLM-4.7-Flash in Tables 8 and 9, respectively. We make three key observations.

First, re-ranking consistently improves recall and accuracy while typically reducing search calls compared to no re-ranking. Notably, gpt-oss-20b with the BM25–monoT5 pipeline (depth 50) achieves the best performance (recall and accuracy) in our study, reaching 0.716 recall and 0.689 accuracy with 27.57 search calls. Relative to no re-ranking, this represents gains of 16.23% in recall and 20.45% in accuracy, alongside a 10.98% reduction in search calls. Despite using only a 20B agent with a BM25 retriever and a 3B re-ranker, this configuration achieves accuracy comparable to a GPT-5–based agent using Qwen3-Embed-8B (0.701; Table 1 in [6]).

Second, no single re-ranker consistently performs best. Interestingly, the reasoning-based re-ranker Rank1 [58] does not show a clear advantage over non-reasoning ones. Table 10 presents a failure case in which Rank1 misinterprets the search intent by incorrectly treating the independent keywords "radiation" and "protein" as a semantic unit, finally leading to a wrong relevance prediction. This suggests that keyword-rich, phrase-driven web-search queries may reduce the effectiveness of Rank1's explicit reasoning. Rank1 is trained on MS MARCO [4] that consists of natural-language questions, resulting in a training–inference query mismatch with agent-issued queries. We examine this issue in detail in Section 4.4.

Third, deeper re-ranking tends to improve effectiveness while reducing search calls. For gpt-oss-20b with the BM25–monoT5 pipeline, increasing the depth from 10 to 20 improves recall and accuracy by 6.21% and 6.81%, respectively, while reducing search calls by 3.11%. Further increasing the depth from 20 to 50 yields gains of 2.14% in recall and 2.23% in accuracy, alongside a 4.80% reduction in search calls. These trends align with [45], which reports improvements with deeper depths for listwise re-rankers in deep research.

4.4 Training–inference query mismatch

To address **RQ3**, we evaluate the gpt-oss-20b agent using BM25, SPLADE-v3, and Qwen3-Embed-8B as retrievers under three search-query conditions: agent-issued queries, and questions generated by two variants of our query-to-question (Q2Q) method (see Section 3.2.5). The two variants generate a question using only the raw query issued by the agent (denoted as Q) or using both the raw query and the agent's recent reasoning trace (denoted as Q+R).

Table 8: Performance of the gpt-oss-20b agent across ranking pipelines with different retrievers, re-rankers, and re-ranking depths on the passage corpus of BrowseComp-Plus. d denotes the re-ranking depth. The best values in recall and accuracy are boldfaced, and the second-best ones are underlined.

Re-ranker	BM25				SPLADE-v3				Qwen3-Embed-8B			
	#Search	Recall	Acc.	Comp.	#Search	Recall	Acc.	Comp.	#Search	Recall	Acc.	Comp.
no re-ranking	30.97	0.616	0.572	0.980	32.75	0.545	0.516	0.980	37.83	0.470	0.417	0.963
monoT5 ($d = 10$)	29.89	0.660	0.631	0.971	31.58	0.630	0.599	0.965	35.64	0.524	0.471	0.969
monoT5 ($d = 20$)	28.96	0.701	0.674	0.960	30.07	0.647	0.598	0.974	34.16	0.570	0.541	0.959
monoT5 ($d = 50$)	27.57	**0.716**	**0.689**	0.974	29.72	0.689	0.646	0.971	32.94	0.614	0.559	0.952
RankLLaMA ($d = 10$)	29.85	0.657	0.617	0.980	31.30	0.632	0.595	0.964	36.07	0.508	0.465	0.964
RankLLaMA ($d = 20$)	29.03	0.681	0.655	0.971	30.82	0.646	0.605	0.981	34.25	0.553	0.494	0.961
RankLLaMA ($d = 50$)	27.10	0.710	0.678	0.977	28.96	0.691	**0.663**	0.959	32.85	0.613	**0.568**	0.964
Rank1 ($d = 10$)	29.62	0.662	0.628	0.966	31.18	0.617	0.580	0.978	35.50	0.530	0.454	0.951
Rank1 ($d = 20$)	28.28	0.694	0.669	0.971	30.73	0.672	0.617	0.953	34.04	0.579	0.528	0.969
Rank1 ($d = 50$)	26.92	0.712	0.687	0.977	29.12	**0.702**	0.643	0.959	32.72	**0.630**	0.564	0.949

Table 9: Performance of the GLM-4.7-Flash (30B) agent on the passage corpus of BrowseComp-Plus. Metrics and formatting follow Table 8.

Retriever	Re-ranker	#Search	Recall	Acc.	Comp.
BM25	no re-ranking	34.73	0.581	0.445	0.863
	monoT5 ($d = 10$)	33.41	0.665	0.549	0.892
	monoT5 ($d = 50$)	31.46	0.696	**0.586**	0.901
	RankLLaMA ($d = 10$)	37.99	0.660	0.503	0.836
	RankLLaMA ($d = 50$)	32.97	**0.707**	0.576	0.865
SPLADE-v3	no re-ranking	36.36	0.578	0.466	0.883
	monoT5 ($d = 10$)	32.24	0.632	0.528	0.865
	monoT5 ($d = 50$)	31.05	0.693	**0.575**	0.900
	RankLLaMA ($d = 10$)	33.97	0.612	0.490	0.880
	RankLLaMA ($d = 50$)	34.62	**0.695**	0.543	0.842

Following Sections 4.2 and 4.3, experiments are performed on the passage corpus. Due to limited GPU resources, we exclude GLM-4.7-Flash. We present results in Table 11.

We make three observations. First, for both retrievers (SPLADE-v3 and Qwen3-Embed-8B), Q2Q (Q+R) consistently achieves significant improvements over raw agent-issued queries. E.g., with SPLADE-v3, Q2Q (Q+R) yields relative gains of 7.34% in recall and 7.95% in accuracy. These results indicate that mismatch between agent-issued queries and the natural-language questions used to train neural rankers severely limits their effectiveness in deep research. Q2Q (Q+R) effectively mitigates this training–inference query mismatch. Second, Q2Q (Q) provides little to no improvement over raw agent-issued queries. To illustrate why, Table 12 illustrates examples of a raw query generated by the gpt-oss-20b agent and its reformulations by Q2Q (Q) and Q2Q (Q+R). The agent's search intent is to retrieve football matches with an attendance of 61,880. The raw query ("61,880" football attendance) is keyword-driven and under-specified. Q2Q (Q) reformulates it as "What is the football attendance number 61,880?", shifting the focus from identifying matches to explaining the number itself. In contrast, Q2Q (Q+R), which incorporates the agent's reasoning trace, better captures the search intent. Third, for BM25, Q2Q-generated questions even hurt performance, indicating that web-search-style queries are better suited to BM25 than natural-language questions.

Given that Q2Q (Q+R) reduces query mismatch for neural retrievers, we further investigate whether it also alleviates mismatch in re-ranking. Section 4.3 shows that keyword- and phrase-based web-search queries can weaken the reasoning effectiveness of the re-ranker Rank1 [58]. We therefore evaluate gpt-oss-20b using a SPLADE-v3–Rank1 pipeline (re-ranking depth $d = 10$). SPLADE-v3 uses raw agent-issued queries, while Rank1 uses the raw queries or the Q2Q (Q+R) reformulations. As shown in Table 13, Q2Q (Q+R) significantly mitigates query mismatch for Rank1, yielding relative gains of 3.40% in recall and 5.69% in accuracy over raw queries.

5 Related Work

Text ranking. Unsupervised lexical retrievers (e.g., BM25 [42]) have long dominated text ranking. With the rise of pre-trained language models (e.g., BERT [10] and T5 [41]) and large-scale human-labelled training data [4], neural rankers have rapidly advanced. A wide range of neural retrievers has been developed, including single-vector dense [59], multi-vector dense [20, 43], and learned sparse retrievers [12, 22]. In addition, cross-encoder re-rankers [38] based on BERT [10] or T5 [41] have achieved strong effectiveness in re-ranking. More recently, LLMs have further advanced neural ranking [31], e.g., through stronger representation capabilities [28] and large-scale synthetic training data [69]. LLM reasoning has also been explored to enhance re-ranking [58, 64, 68]. Text ranking has been studied across diverse settings, including single-hop retrieval [4, 21], multi-hop retrieval [15, 65], reasoning-intensive search [44, 48], and RAG systems [3, 34, 35, 49]. However, little work has systematically examined the performance of text ranking methods in deep research.

Deep research. The origins of deep research can be traced back to multi-hop question answering (QA) [15, 53, 65]. Most multi-hop QA benchmarks rely on Wikipedia, which has been extensively used in LLM pre-training, making such questions less challenging today [6]. In contrast, deep research requires iterative web-scale search and evidence synthesis across the open web [55, 70]. E.g., the BrowseComp benchmark [55] includes queries that humans typically cannot solve within ten minutes using search engines. Solving these queries typically requires LLM-based agents to perform multiple rounds of search and reasoning to gather, synthesise,

Table 10: Example of reasoning trace generated by Rank1 [58], given a search query issued by the gpt-oss-20b agent and one retrieved passage for Query 37 on BrowseComp-Plus. The passage is irrelevant to Query 37. The agent's most recent reasoning trace is: "· · · We will search for 'radiation', 'section 4.1', 'protein', 'two species', and '2010'." Rank1 incorrectly treats the independent keywords "radiation" and "protein" as a coherent semantic unit, ultimately leading to an incorrect prediction.

Search query	Passage	Reasoning text	Score
"section 4.1" radiation protein 2010	· · · A subset panel of significantly altered proteins was selected to build predictive models of radiation exposure and received radiation dose useful for population screening in a future radiological or nuclear event. · · ·	· · · The query mentions "section 4.1" which might refer to a specific section in a document or paper. The term "radiation protein" could be a typo or shorthand for a specific protein related to radiation exposure. · · · The query might be referencing a specific section in a paper titled "radiation protein 2010." Since the passage is from a 2010 study and discusses radiation-related proteins, it's likely relevant. · · ·	0.999

Table 11: Performance of the gpt-oss-20b agent under different retrievers and query conditions on the BrowseComp-Plus passage corpus. Q2Q denotes our query-to-question method (Section 3.2.5); Q uses only the raw agent-issued query, while Q+R additionally incorporates the agent's recent reasoning trace. Within each retriever, the best values in recall and accuracy are boldfaced, and the second-best ones are underlined. * denotes a statistically significant improvement of Q2Q (Q+R) over raw agent-issued queries (paired t-test, $p < 0.05$).

Retriever	Query	#Search	Recall	Acc.	Comp.
BM25	Raw	30.97	**0.616**	0.572	0.968
	Q2Q (Q)	31.88	0.593	**0.578**	0.977
	Q2Q (Q+R)	32.15	0.583	0.557	0.974
SPLADE-v3	Raw	32.75	0.545	0.516	0.980
	Q2Q (Q)	32.88	0.550	0.510	0.976
	Q2Q (Q+R)	31.70	**0.585***	**0.557***	0.983
Qwen3 -Embed-8B	Raw	37.83	0.470	0.417	0.963
	Q2Q (Q)	37.42	0.457	0.404	0.969
	Q2Q (Q+R)	35.82	**0.507***	**0.459***	0.965

Table 12: Examples of search queries issued by the gpt-oss-20b agent, and queries reformulated by the Q2Q method, for Query 781 on BrowseComp-Plus. The agent's recent reasoning trace is: "· · · Let's recall some matches that had an attendance of exactly 61,728, etc. Perhaps it may be easier to search for 'attendance 61,880' 'football.'"

Method	Search query
Raw query	"61,880" football attendance
Q2Q (Q)	What is the football attendance number 61,880?
Q2Q (Q+R)	What football match had an attendance of 61,880?

Table 13: Performance of the gpt-oss-20b agent using a ranking pipeline with the SPLADE-v3 retriever and the Rank1 re-ranker on the BrowseComp-Plus passage corpus. SPLADE-v3 always uses raw agent-issued queries, while Rank1 operates under different query conditions. Metric definitions and formatting follow Table 11.

Re-ranker	Query	#Search	Recall	Acc.	Comp.
Rank1 ($d = 10$)	Raw	31.18	0.617	0.580	0.978
	Q2Q (Q+R)	31.15	**0.638***	**0.613***	0.970

and verify evidence [39, 50]. Most existing approaches equip LLM-based agents with live web search APIs [23, 54], but such black-box systems lack transparency. To address this issue, Chen et al. [6] curate BrowseComp-Plus, which extends the original BrowseComp dataset with a fixed document corpus and human-verified relevance judgments, allowing white-box retrievers to be used. On this resource, Sharifymoghaddam and Lin [45] study listwise LLM-based re-rankers to analyse effectiveness–efficiency trade-offs. We differ from prior studies [6, 45] by systematically reproducing a broad spectrum of text ranking methods in deep research. Unlike contemporaneous work [17] that tests BM25 and LLM-based single-vector retrievers in scientific literature search (domain-specific deep research), we use a broader range of retrievers (e.g., learned sparse and multi-vector) and re-rankers in a general open-domain setting.

6 Conclusions & Future Work

We have reproduced an extensive set of text ranking methods in deep research, conducting experiments on BrowseComp-Plus [6] with 2 open-source agents, 5 retrievers, and 3 re-rankers.

Overall, our results show that text ranking methods developed by the IR community are still highly effective in deep research. Several well-established findings continue to hold: (i) the lexical retriever BM25 with appropriate setup outperforms neural rankers in most cases; notably, gpt-oss-20b with BM25 on the passage corpus achieves the highest answer accuracy across all retrieval settings in our study; (ii) BERT-based learned sparse [22] and multi-vector dense [43] retrievers generalise better than LLM-based single-vector dense retrievers; and (iii) re-ranking remains highly effective, improving recall and answer accuracy while reducing search calls, with deeper re-ranking depths further amplifying these gains.

However, the web-search-style syntax of agent-issued queries (e.g., quoted exact matches) induces distribution drift for neural rankers, limiting their effectiveness in deep research and preventing a prior finding from generalising to this setting: the reasoning-based re-ranker Rank1 [58] often misinterprets such queries, diminishing the benefits of reasoning and showing no clear advantage over non-reasoning methods. Our proposed Q2Q method significantly mitigates this drift, improving neural ranking performance.

Beyond validating these findings, we find that passage-level units benefit agents with limited context windows, and reduce sensitivity to document-length normalisation in lexical retrieval.

We identify future directions: (i) we only use two LLMs as agents; evaluating additional model families and larger model sizes would help assess the generalisability of our findings; (ii) exploring additional rankers [2, 25, 44] and configurations, such as scaling laws, is an important direction; and (iii) extending deep research to conversational scenarios [29, 32–34] is a promising avenue.

Acknowledgments

We thank Zijian Chen and Xueguang Ma for their guidance on using BrowseComp-Plus. This research was supported by a Turing AI Acceleration Fellowship funded by the Engineering and Physical Sciences Research Council (EPSRC), grant number EP/V025708/1.

References

[1] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. 2025. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925* (2025).

[2] Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. 2026. jina-embeddings-v5-text: Task-Targeted Embedding Distillation. *arXiv preprint arXiv:2602.15547* (2026).

[3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *ICLR*.

[4] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268* (2018).

[5] Antoine Chaffin and Raphaël Sourty. 2025. PyLate: Flexible Training and Retrieval for Late Interaction Models. In *CIKM*. 6334–6339.

[6] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. 2025. BrowseComp-Plus: A More Fair and Transparent Evaluation Benchmark of Deep-Research Agent. *arXiv preprint arXiv:2508.06600* (2025).

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2019. Overview of the TREC 2019 Deep Learning Track. In *REC 2019*.

[8] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *SIGIR*. 985–988.

[9] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2021. TREC CAsT 2021: The Conversational Assistance Track Overview. In *TREC*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[11] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[12] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR*. 2288–2292.

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948* (2025).

[14] Ben He and Iadh Ounis. 2005. Term Frequency Normalisation Tuning for BM25 and DFR Models. In *ECIR*. 200–214.

[15] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *COLING*. 6609–6625.

[16] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

[17] Tiansheng Hu, Yilun Zhao, Canyu Zhang, Arman Cohan, and Chen Zhao. 2026. SAGE: Benchmarking and Improving Retrieval for Deep Research Agents. *arXiv preprint arXiv:2602.05975* (2026).

[18] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. In *COLM*.

[19] Marcin Kaszkiel and Justin Zobel. 1997. Passage Retrieval Revisited. In *ACM SIGIR Forum*, Vol. 31. ACM New York, NY, USA, 178–185.

[20] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*. 39–48.

[21] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural Questions: A Benchmark for Question Answering Research. *TACL* 7 (2019), 453–466.

[22] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. SPLADE-v3: New baselines for SPLADE. *arXiv preprint arXiv:2403.06789* (2024).

[23] Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. 2025. WebSailor: Navigating Super-human Reasoning for Web Agent. *arXiv preprint arXiv:2507.02592* (2025).

[24] 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. *arXiv preprint arXiv:2501.05366* (2025).

[25] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards General Text Embeddings with Multi-stage Contrastive Learning. *ArXiv* (2023).

[26] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2022. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature.

[27] Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, et al. 2025. WebExplorer: Explore and Evolve for Training Long-Horizon Web Agents. *arXiv preprint arXiv:2509.06501* (2025).

[28] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-Tuning LLaMA for Multi-Stage Text Retrieval. In *SIGIR*. 2421–2425.

[29] Chuan Meng, Mohammad Aliannejadi, and Maarten de Rijke. 2023. System Initiative Prediction for Multi-turn Conversational Information Seeking. In *CIKM*. 1807–1817.

[30] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Ranked List Truncation for Large Language Model-based Re-Ranking. In *SIGIR*. 141–151.

[31] Chuan Meng, Jiqun Liu, Mohammad Aliannejadi, Fengran Mo, Jeff Dalton, and Maarten de Rijke. 2026. Re-Rankers as Relevance Judges. *arXiv preprint arXiv:2601.04455* (2026).

[32] Chuan Meng, Fengran Mo, Mohammad Aliannejadi, Jeff Dalton, and Jian-Yun Nie. 2026. Conversational Search: Foundations, Large Language Models, and Agents. In *ECIR*. 11–18.

[33] Chuan Meng, Francesco Tonolini, Fengran Mo, Nikolaos Aletras, Emine Yilmaz, and Gabriella Kazai. 2025. Bridging the Gap: From Ad-hoc to Proactive Search in Conversations. In *SIGIR*. 64–74.

[34] Fengran Mo, Yifan Gao, Chuan Meng, Xin Liu, Zhuofeng Wu, Kelong Mao, Zhengyang Wang, Pei Chen, Zheng Li, Xian Li, et al. 2025. UniConv: Unifying Retrieval and Response Generation for Large Language Models in Conversations. In *ACL*. 6936–6949.

[35] Fengran Mo, Zhan Su, Yuchen Hui, Jinghan Zhang, Jia Ao Sun, Zheyuan Liu, Chao Zhang, Tetsuya Sakai, and Jian-Yun Nie. [n. d.]. OpenDecoder: Open Large Language Model Decoding to Incorporate Document Quality in RAG. In *WWW*. 2252–2262.

[36] Ariane Mueller and Craig Macdonald. 2025. Semantically Proportioned nDCG for Explaining ColBERT's Learning Process. In *ECIR*. 341–356.

[37] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[38] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *EMNLP*. 708–718.

[39] Litu Ou, Kuan Li, Huifeng Yin, Liwen Zhang, Zhongwang Zhang, Xixi Wu, Rui Ye, Zile Qiao, Pengjun Xie, Jingren Zhou, and Yong Jiang. 2025. BrowseConf: Confidence-Guided Test-Time Scaling for Web Agents. *arXiv preprint arXiv:2510.23458* (2025).

[40] Paul Owoicho, Jeffrey Dalton, Mohammad Aliannejadi, Leif Azzopardi, Johanne R. Trippas, and Svitlana Vakulenko. 2022. TREC CAsT 2022: Going Beyond User Ask and System Retrieve with Initiative and Response Generation. In *TREC*.

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[42] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *TREC*.

[43] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *NAACL*. 3715–3734.

[44] Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, et al. 2025. ReasonIR: Training Retrievers for Reasoning Tasks. *arXiv preprint arXiv:2504.20595* (2025).

[45] Sahel Sharifymoghaddam and Jimmy Lin. 2026. Rerank Before You Reason: Analyzing Reranking Tradeoffs through Effective Token Cost in Deep Search Agents. *arXiv preprint arXiv:2601.14224* (2026).

[46] Zhengliang Shi, Yiqun Chen, Haitao Li, Weiwei Sun, Shiyu Ni, Yougang Lyu, Run-Ze Fan, Bowen Jin, Yixuan Weng, Minjun Zhu, et al. 2025. Deep Research: A Systematic Survey. *arXiv preprint arXiv:2512.02038* (2025).

[47] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-Searcher: Incentivizing the Search Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2503.05592* (2025).

[48] Hongjin SU, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Liu Haisu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O Arik, Danqi Chen, and Tao Yu. 2025. BRIGHT: A Realistic and Challenging Benchmark for Reasoning-Intensive Retrieval. In *ICLR*.

[49] Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. 2025. Parametric Retrieval Augmented Generation. In *SIGIR*. 1240–1250.

[50] 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, Xixi Wu, Xuanzhong Chen, Yida Zhao, Zhen Zhang, Zhengwei Tao, Zhongwang Zhang, Zile Qiao, Chenxi Wang, Donglei Yu, Gang Fu, Haiyang Shen, Jiayin Yang, Jun Lin, Junkai Zhang, Kui Zeng, Li Yang, Hailong Yin, Maojia Song, Ming Yan, Minpeng Liao, Peng Xia, Qian Xiao, Rui Min, Ruixue Ding, Runnan Fang, Shaowei Chen, Shen Huang, Shihang Wang, Shihao Cai, Weizhou Shen, Xiaobin Wang, Xin Guan, Xinyu Geng, Yingcheng Shi, Yuning Wu, Zhuo Chen, Zijian Li, and Yong Jiang. 2025. Tongyi DeepResearch Technical Report. *arXiv preprint arXiv:2510.24701* (2025).

[51] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288* (2023).

[53] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *TACL* 10 (2022), 539–554.

[54] Yibo Wang, Lei Wang, Yue Deng, Keming Wu, Yao Xiao, Huanjin Yao, Liwei Kang, Hai Ye, Yongcheng Jing, and Lidong Bing. 2026. DeepResearchEval: An Automated Framework for Deep Research Task Construction and Agentic Evaluation. *arXiv preprint arXiv:2601.09688* (2026).

[55] 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 preprint arXiv:2504.12516* (2025).

[56] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research* (2022).

[57] Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2025. On the Theoretical Limitations of Embedding-Based Retrieval. *arXiv preprint arXiv:2508.21038* (2025).

[58] Orion Weller, Kathryn Ricci, Eugene Yang, Andrew Yates, Dawn Lawrie, and Benjamin Van Durme. 2025. Rank1: Test-Time Compute for Reranking in Information Retrieval. In *COLM*.

[59] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *ICLR*.

[60] Fangyuan Xu, Rujun Han, Yanfei Chen, Zifeng Wang, I Hsu, Jun Yan, Vishy Tirumalashetty, Eunsol Choi, Tomas Pfister, Chen-Yu Lee, et al. 2026. SAGE: Steerable Agentic Data Generation for Deep Search with Execution Feedback. *arXiv preprint arXiv:2601.18202* (2026).

[61] Yilong Xu, Zhi Zheng, Xiang Long, Yujun Cai, and Yiwei Wang. 2026. Self-Manager: Parallel Agent Loop for Long-form Deep Research. *arXiv preprint arXiv:2601.17879* (2026).

[62] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388* (2025).

[63] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).

[64] Eugene Yang, Andrew Yates, Kathryn Ricci, Orion Weller, Vivek Chari, Benjamin Van Durme, and Dawn Lawrie. 2025. Rank-K: Test-Time Reasoning for Listwise Reranking. *arXiv preprint arXiv:2505.14432* (2025).

[65] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*. 2369–2380.

[66] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*.

[67] Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. 2025. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *arXiv preprint arXiv:2508.06471* (2025).

[68] Jinghan Zhang, Xiting Wang, Fengran Mo, Yeyang Zhou, Wanfu Gao, and Kunpeng Liu. 2025. Entropy-based Exploration Conduction for Multi-step Reasoning. In *ACL*. 3895–3906.

[69] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176* (2025).

[70] Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, et al. 2025. BrowseComp-ZH: Benchmarking Web Browsing Ability of Large Language Models in Chinese. *arXiv preprint arXiv:2504.19314* (2025).

[71] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the Gap Between Indexing and Retrieval for Differentiable Search Index with Query Generation. *arXiv preprint arXiv:2206.10128* (2022).