

LLM-based Query Expansion Under Control: Reproducibility, Portability, and Cost

Pouya Sadeghi
University of Waterloo
Waterloo, Ontario, Canada
pouya.sadeghi@uwaterloo.ca

Jimmy Lin
University of Waterloo
Waterloo, Ontario, Canada
jimmylin@uwaterloo.ca

Abstract

Query expansion with large language models (LLMs) can fill the gap in lexical search when users and relevant documents describe the same concepts in different words. Published gains are difficult to compare because studies use different LLMs, retrieval systems, and evaluation protocols. We introduce Expandinini, a framework for cost-aware evaluation of LLM-based query expansion under controlled retrieval and evaluation settings. We compare 13 LLM-based query expansion methods across 22 LLM configurations and 25 collections spanning TREC Deep Learning, BEIR, and BRIGHT. Our implementations improve over BM25 in most TREC DL comparisons and stay consistent with the gains reported in the original studies. Using a larger model does not always lead to gains in effectiveness; we observe no meaningful average improvement from Qwen2.5-32B to 72B on our six primary collections. Our ablations show that combining multiple expansions and generating passages or answers instead of keywords are beneficial, while additional iterations provide limited gains. In our cost comparison, ADORE performs best on average, while MuGI offers most of ADORE’s gain at a much lower cost. Post-retrieval methods lead on most collections. Expandinini provides infrastructure for benchmarking retrieval effectiveness against inference cost across backbones and helps practitioners assess whether larger models or multi-round pipelines justify their cost. We release Expandinini along with commands to reproduce our experiments at <https://github.com/castorini/expandinini>.

CCS Concepts

• Information systems → Query reformulation; Evaluation of retrieval results.

Keywords

query expansion, large language models, reproducibility, retrieval evaluation, cost-effectiveness

ACM Reference Format:

Pouya Sadeghi and Jimmy Lin. 2026. LLM-based Query Expansion Under Control: Reproducibility, Portability, and Cost. In *Proceedings of the 2026 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP 2026)*, December 13–15, 2026, Singapore, Singapore. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3848000.3850204>



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
SIGIR-AP 2026, Singapore, Singapore
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3033-7/2026/12
<https://doi.org/10.1145/3848000.3850204>

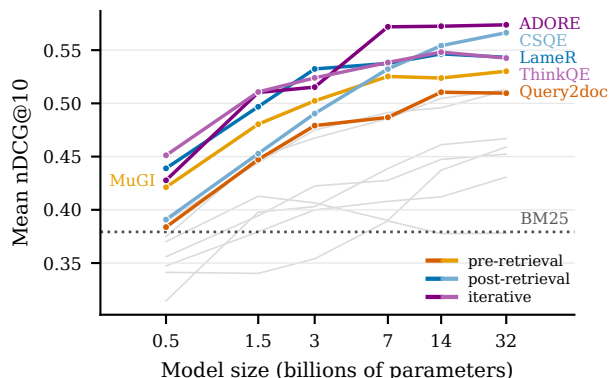


Figure 1: Mean nDCG@10 on DL19/20/22/23 for 13 methods across the Qwen2.5 models from 0.5B to 32B. The dotted line marks the BM25 baseline. Model size changes both scores and rankings, and only six methods outperform BM25 on average across the four DL collections at 0.5B.

1 Introduction

Lexical retrieval matches a query to documents through the terms they share, and its long-standing weakness is that the words a searcher uses may not be the words an author wrote [16, 37]. Query expansion addresses this gap by adding related terms to the query. For decades, those terms came from the collection itself, through pseudo-relevance feedback that assumes the top-ranked documents are relevant and mines expansion terms from them [1, 38]. This helps when the initial ranking contains useful feedback, but irrelevant documents can pull the expansion off topic. Moreover, feedback documents may lack the vocabulary needed to retrieve relevant documents outside that ranking. On the other hand, large language models (LLMs) can generate expansion terms or pseudo-documents directly from a query, without relying on retrieved documents.

Comparing the gains reported by LLM-based expansion methods is difficult because choices outside the method affect the score. Studies start from different implementations of BM25, itself a family of scoring variants rather than one system [21]. They binarize graded relevance judgments differently, which changes average precision and recall. They also use different LLMs to generate expansions, which leaves it unclear whether a method remains effective with a smaller LLM. Expansion cost is often omitted, despite its importance to evaluating natural language processing and information retrieval systems [12, 39]. When the baseline, the metric, the LLM, and the budget all vary, a gain is hard to attribute and to act on.

In this work, we study LLM-based query expansion methods on 25 collections using a fixed BM25 retriever, without dense or hybrid retrieval or reranking. We compare 13 expansion methods across LLM configurations and collections. By holding the retrieval and evaluation fixed, we isolate the query-side contribution of the LLM, which allows us to assess how the choice of LLM affects expansion effectiveness and cost. We investigate four research questions:

- RQ1* How closely do our implementations reproduce the results reported in the original studies?
- RQ2* How do model size and family affect the absolute and relative effectiveness of query expansion methods?
- RQ3* What are the trade-offs between effectiveness and cost across methods, and how does reasoning effort affect them?
- RQ4* How do expansion methods generalize across collections?

To answer these questions, we build Expandorini, a reproducible evaluation framework that standardizes LLM-based query expansion, retrieval, and cost accounting behind a shared interface, and we use it to implement and evaluate 13 representative expansion methods. It uses Lucene BM25 over local Pyserini indexes [29], checks baseline scores against available community reference results [49], and records each collection’s evaluation conventions. Each method’s implementation and prompt templates remain fixed across API and open-weight models (§3). Each run records token counts, metered API cost, or estimated generation FLOPs. We consider only generation compute, which is the LLM computation used for query expansion, and exclude retrieval computation.

The study covers 25 collections and 22 configurations of 18 models, comprising 7,150 combinations, each evaluated over three runs, and 550 BM25 controls. Six primary collections support the main analyses, while 19 additional collections test whether the findings extend to other domains. We vary model size within the Qwen2.5 family and examine other model families and reasoning budgets separately (§4). Figure 1 shows how effectiveness and method rankings change across Qwen2.5 models from 0.5B to 32B on the four TREC DL collections. Our experiments support four findings:

- F1* In almost all cases, our implementations show gains over BM25, consistent with the original studies. The exception is ReFormeR on DL19.
- F2* Larger models do not always improve effectiveness. In our experiments, we do not see a meaningful average improvement when moving from Qwen2.5-32B to 72B.
- F3* In our cost comparison, ADORE performs best on average, but requires the most generation compute. MuGI offers most of ADORE’s gain at a much lower cost.
- F4* Overall, post-retrieval methods, particularly iterative methods, are the strongest query expansion methods in our comparisons across 25 collections.

We release Expandorini and our artifacts to support reproducibility and further analysis: <https://github.com/castorini/expandorini>.

2 Related Work

LLM-based query expansion. LLM-based query expansion methods can be grouped into main pre-retrieval and post-retrieval approaches [50]. *Pre-retrieval* methods generate expansions from the query without using retrieved documents and by only relying on the

LLM’s parametric knowledge learned during training. These methods include keyword and chain-of-thought prompting [20], pseudo-document generation [17, 46], combining multiple generated expansions [51], using different prompting instructions [11, 47], and dialogue between LLM agents [40]. *Post-retrieval* methods use documents retrieved from the corpus for the query to guide expansion through pseudo-relevance feedback [1, 38]. These methods expand the query using retrieved passages [41], extract corpus evidence to guide generation [26], or apply mined reformulation patterns [5]. Some post-retrieval methods alternate retrieval and query refinement over multiple rounds, using feedback from retrieved documents to guide subsequent expansions. We refer to these approaches as *iterative* methods. Examples of this group include ThinkQE [27], which combines a thinking-based expansion process with corpus interaction, and ADORE [2], which uses LLM relevance judgments to guide query refinement across rounds. Given the growing interest in iterative expansion and the effectiveness reported by these methods, we treat them as a third group, separate from the other post-retrieval methods. Methods in all three groups report gains over their respective baselines. Table 1 records our implementation choices for the 13 methods in this study. Document-side expansion [32] and learned sparse retrieval [15] instead modify document representations and fall outside our query-side comparison.

Comparative evaluations. Weller et al. [48] evaluate 11 techniques over 24 retrievers and observe more gains where the retriever is weak. QueryGym [3] provides a shared interface for query reformulation methods, while Bigdeli et al. [4] compare ten methods across three retrieval paradigms and find that gains depend on the paradigm. We hold the retriever fixed and compare expansion methods across different LLMs. These experiments cover seven model sizes within Qwen2.5, additional model families, and different reasoning budgets. We also repeat each experiment to measure variation in retrieval scores and use matched ablations to examine the expansion choices that published methods combine.

Reproducibility and cost. Anserini [49], Pyserini [29], and two-click reproduction [28] support reproducible retrieval experiments through shared implementations, reference results, and documented commands. Kamphuis et al. [21] show why the retrieval configuration matters even when systems use the same BM25 label. We build on these works by recording retrieval settings, relevance conventions, prompts, and regeneration commands alongside the results. Following calls to report computational cost in NLP [12] and IR [39], we record metered API spending and estimate generation FLOPs for open-weight models [22].

3 The Expandorini Framework

Expandorini is a modular evaluation framework that decouples expansion generation from the underlying retrieval and evaluation pipeline. Through a shared interface, any method can generate expansions from an input query, conditioning on retrieved documents or iterating across multiple retrieval rounds when required. This design lets us compare methods and LLM configurations under the same retrieval and evaluation settings for each collection and record these settings for reproducibility.

Retrieval and Baselines. All methods use collection-specific indexes and retrieval settings through Pyserini’s [29] implementation of Lucene BM25 to retrieve the top $k = 1000$ documents per query. As our main baseline, we use BM25 with the unexpanded query, which is indicated as `direct` in the run records. We also evaluate RM3 [1] as a pseudo-relevance feedback baseline.

The unexpanded BM25 scores match the published reference values to four decimal places on seven collections. We compare against the two-click reproduction (2CR) results [28] for DL19, DL20, SciFact, and ArguAna, and the Anserini regression results for DL22, DL23, and RAGgy-dev. These comparisons use the collection-specific evaluation protocols, as described below.

Evaluation Protocols. We follow each collection’s evaluation conventions for relevance thresholds, metric cutoffs, and document exclusions. For example, TREC DL passage evaluation counts grades 2 and 3 as relevant for binary metrics while retaining all grades for nDCG [29]. RAGgy-dev counts grades 1 and above as relevant and evaluates MAP and MRR at a cutoff of 100 [49]. We also remove self-matches on ArguAna [45] and apply BRIGHT’s per-query document exclusions before the evaluation cutoff [43].

Expansion Generation. Every method shares a fixed implementation and prompt template across backbones, routed through a uniform OpenAI-compatible interface via vLLM [24] for open-weight models and `pi` [13], with tools, sessions, and extensions disabled, for commercial APIs. Post-retrieval and iterative methods, such as LameR [41], CSQE [26], ReFormer [5], ThinkQE [27], and ADORE [2], which require corpus feedback, use the top 10 passages retrieved by BM25 at each round, if multiple, to guide the expansion generation. For API runs, we explicitly log decoding parameters and vary reasoning effort (§7). For open-weight models that support thinking mode, we disable it throughout.

Generation Cost. We record input and output token counts for every run, along with metered spending for API models. For open-weight models, we estimate generation FLOPs as $2N$ times the total number of input and output tokens, where N is the model’s parameter count [22]. For mixture-of-experts models, we use the active parameter count. Our cost estimates cover LLM inference for query expansion and exclude retrieval computation. The API experiments cost about \$12,300, with GPT-5.5 accounting for roughly 60% of that. Token counts and costs are included in the run records.

4 Experimental Setup

Methods and implementation. We evaluate 13 LLM-based query expansion methods in an inference-only setting without any task-specific fine-tuning. Where an original method relies on dense retrieval or a learned post-processor, we adapt its expansion generator to run over BM25. Table 1 summarizes our implementation decisions alongside the baselines. We follow the published method description by default and document differences from the original paper or released code. For ADORE [2], we accumulate feedback across rounds as described in the paper, rather than using only the previous round’s evidence as in the released code. For ReFormer [5], we evaluate the released two-call architecture without the supervised selector described in the paper. For MuGI [51], we use the released code’s query repetition factor of 6, compared with 4 in

Table 1: Implementation details for baselines and query expansion methods, grouped by expansion approach. Rows note differences from the original papers or released code.

Method	Implementation notes
BM25 (direct) [37]	Matches reference scores to four decimal places.
RM3 [1]	Uses Pyserini’s default RM3 settings.
Q2E [20]	Uses the original prompt without Terrier weighting.
CoT [20]	Keeps the answer, which is removed in the paper.
Query2doc [46]	Uses zero-shot prompting instead of the four-shot default.
HyDE [17]	Uses one generated pseudo-document with BM25. The original method uses dense retrieval with eight samples. Prompts are reconstructed because none are published. Only the sparse retrieval variant is evaluated.
AMD [40]	Prompts are reconstructed because none are published. Only the sparse retrieval variant is evaluated.
GenQR [47]	Generates five samples; omits the Terrier weighting.
GenQREnsemble [11]	Uses the original prompt with different decoding settings.
MuGI [51]	Prompt unchanged; repeats the query six times as in the released code, rather than four times as in the paper.
LameR [41]	Prompt unchanged; does not apply the original 128-token truncation to the query and retrieved context.
CSQE [26]	Prompt unchanged; two query repetitions instead of four.
ReFormer [5]	Released two-call architecture; no supervised selector.
ThinkQE [27]	Prompt unchanged; context is truncated differently.
ADORE [2]	Accumulates feedback across rounds as in the paper. The released code uses only the previous round’s feedback.

the paper. For LameR [41], we do not apply the original 128-token truncation of the query and context.

Collections. We evaluate on 25 collections, with six *primary collections* supporting the main analyses. DL19 and DL20 [7, 9] use MS MARCO v1 passages [31] and contain 43 and 54 judged queries, respectively. DL22 and DL23 [8, 10] use MS MARCO v2 passages and contain 76 and 82 judged queries. RAGgy-dev [34] uses v2.1 segmented documents with 120 queries, and SciFact [45] contains 300 queries. The remaining 19 collections include DL-Hard [30], six BEIR [45] collections (TREC-COVID, FiQA, NFCorpus, DBpedia-entity, ArguAna, and TREC-News), and all 12 BRIGHT subsets [43]. Each collection uses the evaluation conventions described in §3.

LLM backbones. We use 18 models with a total of 22 configurations, where a configuration specifies the model and, where applicable, its reasoning effort. To study model size within one family, we evaluate Qwen2.5-Instruct [44] at 0.5B, 1.5B, 3B, 7B, 14B, 32B, and 72B. We also evaluate Qwen3.5-9B [35] with thinking disabled and Qwen3.6-35B-A3B [36], a mixture-of-experts model with 35B total and 3B active parameters. The main API configurations are GPT-5.5 at low reasoning and GPT-5.4-mini at low, medium, and high reasoning. Additional configurations include GPT-5.6-Luna at low, medium, and high reasoning, Gemma-4 at E4B, 12B, and 31B, and Gemma-2 at 2B, 9B, and 27B. All listed configurations are evaluated with the full method set on all 25 collections.

Experimental design. Our experiments pair 13 methods with 25 collections and 22 LLM configurations, giving $22 \times 25 \times 13 = 7,150$ combinations. For open-weight and API models, we evaluate each combination with three random seeds and report mean effectiveness across the three runs. This is 21,450 evaluations in total, excluding

Table 2: Our implementations improve over BM25 in 19 of 20 TREC DL comparisons. The exception is ReFormeR on DL19. Each row reports nDCG@10 as baseline → expansion (gain). Q2E and CoT report no TREC DL results.

Method	Original LLM	Our LLM	DL19		DL20	
			Paper	Ours	Paper	Ours
Query2doc	text-davinci-003	GPT-5.5	0.512→0.662 (+.150)	0.506→0.673 (+.167)	0.477→0.629 (+.152)	0.480→0.640 (+.160)
HyDE	text-davinci-003	GPT-5.5	0.445→0.613 (+.168)	0.506→0.675 (+.169)	0.421→0.579 (+.158)	0.480→0.648 (+.168)
GenQR	Flan-T5-XXL	Qwen2.5-14B	0.517→0.556 (+.039)	0.506→0.509 (+.003)	–	–
GenQREnsemble	Flan-T5-XXL	Qwen2.5-14B	0.480→0.564 (+.084)	0.506→0.563 (+.057)	–	–
MuGI	GPT-4-Turbo-1106	GPT-5.5	0.506→0.704 (+.198)	0.506→0.682 (+.176)	0.480→0.644 (+.164)	0.480→0.633 (+.153)
LameR	gpt-3.5-turbo	GPT-5.5	0.506→0.691 (+.185)	0.506→0.667 (+.161)	0.480→0.648 (+.168)	0.480→0.660 (+.180)
CSQE	gpt-3.5-turbo-0301	GPT-5.5	0.506→0.673 (+.167)	0.506→0.688 (+.182)	0.480→0.662 (+.182)	0.480→0.652 (+.172)
AMD	Qwen2.5-7B	Qwen2.5-7B	0.424→0.587 (+.163)	0.506→0.541 (+.035)	0.476→0.582 (+.106)	0.480→0.493 (+.013)
ThinkQE	R1-Distill-Qwen-14B	Qwen2.5-14B	0.506→0.688 (+.182)	0.506→0.689 (+.183)	0.480→0.647 (+.167)	0.480→0.648 (+.168)
ReFormeR	Qwen2.5-7B	Qwen2.5-7B	0.497→0.544 (+.047)	0.506→0.488 (−.018)	0.488→0.572 (+.084)	0.480→0.517 (+.037)
ADORE	GPT-4.1	GPT-5.5	0.506→0.713 (+.207)	0.506→0.715 (+.209)	0.480→0.712 (+.232)	0.480→0.694 (+.214)

BM25 and RM3 baselines and separate ablations. Our primary analysis evaluates all 13 expansion methods on the six primary collections using the nine Qwen configurations, GPT-5.5 at low reasoning effort, and GPT-5.4-mini at low, medium, and high reasoning effort. These 13 LLM configurations provide 78 collection–configuration pairs for comparisons against BM25.

Statistical analysis. We compare each method against BM25 using paired randomization tests with 10,000 permutations [42], applying Holm correction across the 13 methods within each collection–configuration pair. To measure variation across runs, we calculate the standard deviation across the three runs for the nine Qwen configurations on DL19, DL20, DL22, and DL23. This covers 468 method–collection–configuration combinations, with three runs per combination including the original run.

5 RQ1: Reproduction Results

We compare our implementations with published DL19 and DL20 results for 11 methods, using the LLMs listed in Table 2. Q2E and CoT [20] report no results on either collection and are excluded from this comparison. For each available method–collection pair, we compare the nDCG@10 gain over our unexpanded BM25 baseline with the gain over the original study’s baseline. Our implementations improve over BM25 in 19 of 20 comparisons, except for ReFormeR on DL19.

To remain close to the original paper, in Table 2, we use the original LLM for AMD and ReFormeR and substitute other LLMs for the remaining methods. We report implementation differences in Table 1. Retrieval settings also differ across original studies and can affect the measured gains [21]. In particular, HyDE’s [17] published results use dense retrieval with Contriever [19], whereas our implementation uses its generated pseudo-documents with BM25.

Several methods achieve gains close to those originally reported. ThinkQE’s [27] gains differ by at most 0.001 nDCG@10 on DL19 and DL20, although we replace its reasoning model with Qwen2.5-14B. ADORE’s [2] gain on DL19 is 0.002 higher than reported. The gains for LameR [41], CSQE [26], MuGI [51], and zero-shot Query2doc [46] differ from their published values by at most 0.024.

Other methods improve over BM25 but achieve smaller gains than reported. GenQR [47] and GenQREnsemble [11] have gains about 0.03 smaller than their published values. For both methods, our LLM and index configuration differ from those in the original studies. AMD’s [40] gains fall from +0.163 to +0.035 on DL19 and from +0.106 to +0.013 on DL20. Its published BM25 baseline is lower than ours on DL19, whereas the DL20 baselines are similar. We evaluate only AMD’s sparse retrieval variant.

For ReFormeR [5], we use the original LLM and implement the released two-call architecture without the supervised selector described in the paper (§4). On DL19, this implementation scores below our BM25 baseline (0.4877 versus 0.5058), whereas the original paper reports a gain of +0.047. As an implementation check, we run QueryGym’s [3] unmodified implementation (commit 6d2d192). It scores 0.435 nDCG@10 on DL19 and 0.471 on DL20.

6 RQ2: Portability Across LLMs

Larger models do not always improve effectiveness. We examine how absolute and relative effectiveness of query expansion methods change across LLMs, then use controlled ablations to study the expansion choices behind these results.

6.1 Effectiveness Against BM25

Table 3 reports two methods from each group across 10 LLM configurations. ADORE [2], ThinkQE [27], and LameR [41] achieve the highest average nDCG@10 across the 13 main LLM configurations and six primary collections. ADORE ranks first in 43 of the 78 collection–configuration pairs, ThinkQE in 17, and LameR in nine. GenQR [47] and ReFormeR [5] have the lowest averages, where ReFormeR scores 0.4402 against 0.4365 for BM25.

We compare each method against BM25 using paired randomization tests on per-query nDCG@10 scores [42], with 10,000 permutations. We apply Holm’s correction [18] to the 13 comparisons within each collection–configuration pair (Table 5). Nine of the 13 methods significantly outperform BM25 in more than half of the 78 pairs. ThinkQE [27] significantly outperforms BM25 in 74, LameR [41] in 72, and ADORE [2] in 67. GenQR [47] records 20 significant improvements and two declines, AMD [40] records 27

Table 3: A stable top tier persists across model families, with ADORE or ThinkQE leading 70 of the 80 combinations of LLM configuration, metric, and benchmark group. Mean nDCG@10 (black) and R@1k (grey) are reported for two methods per group (pre-retrieval, post-retrieval, iterative) across 10 LLM backbones, grouped by benchmark category. Each block’s grey header gives its BM25 baseline scores. Bold indicates the best score for each backbone and metric within each benchmark block.

Method		Qwen										Gemma-4				GPT								
		2.5-7B		2.5-14B		2.5-32B		2.5-72B		3.6-35B-A3B		12B		31B		5.4-mini _[low]		5.5 _[low]		5.6-Luna _[low]				
TREC DL DL19 / DL20 / DL22 / DL23 (macro)																					BM25: 0.379 0.576			
Pre-retrieval	Query2doc	0.488	0.727	0.503	0.718	0.517	0.736	0.532	0.741	0.536	0.740	0.497	0.708	0.536	0.736	0.522	0.714	0.541	0.731	0.529	0.714			
	MuGI	0.526	0.733	0.515	0.744	0.534	0.757	0.544	0.757	0.551	0.756	0.515	0.723	0.545	0.743	0.550	0.750	0.565	0.771	0.540	0.742			
Post-retrieval	LameR	0.531	0.731	0.544	0.739	0.564	0.755	0.522	0.733	0.544	0.750	0.444	0.660	0.565	0.734	0.577	0.735	0.573	0.748	0.543	0.736			
	CSQE	0.532	0.727	0.568	0.744	0.565	0.756	0.539	0.742	0.557	0.745	0.546	0.731	0.561	0.751	0.544	0.731	0.562	0.750	0.544	0.729			
Iterative	ThinkQE	0.538	0.726	0.554	0.736	0.551	0.741	0.548	0.743	0.570	0.753	0.552	0.742	0.564	0.746	0.600	0.763	0.594	0.762	0.581	0.751			
	ADORE	0.571	0.747	0.568	0.757	0.570	0.761	0.586	0.759	0.583	0.758	0.599	0.741	0.580	0.738	0.607	0.758	0.616	0.781	0.577	0.732			
TREC-RAGgy 2024 RAGgy-dev; 120-topic development set																					BM25: 0.423 0.574			
Pre-retrieval	Query2doc	0.426	0.514	0.443	0.522	0.469	0.531	0.461	0.532	0.440	0.529	0.436	0.506	0.452	0.537	0.491	0.533	0.485	0.543	0.495	0.540			
	MuGI	0.433	0.522	0.455	0.532	0.458	0.533	0.460	0.552	0.458	0.545	0.435	0.515	0.467	0.549	0.460	0.531	0.473	0.556	0.457	0.531			
Post-retrieval	LameR	0.502	0.532	0.497	0.548	0.510	0.545	0.494	0.541	0.493	0.548	0.469	0.518	0.513	0.516	0.502	0.531	0.487	0.535	0.448	0.501			
	CSQE	0.471	0.516	0.478	0.526	0.484	0.535	0.486	0.535	0.471	0.536	0.476	0.527	0.476	0.536	0.488	0.526	0.494	0.543	0.478	0.528			
Iterative	ThinkQE	0.518	0.547	0.518	0.553	0.534	0.553	0.514	0.547	0.518	0.550	0.513	0.549	0.515	0.558	0.540	0.557	0.529	0.570	0.515	0.560			
	ADORE	0.478	0.541	0.490	0.545	0.504	0.555	0.485	0.555	0.487	0.558	0.487	0.529	0.501	0.525	0.510	0.548	0.511	0.562	0.490	0.527			
BEIR ArguAna / DBpedia / FiQA / NFCorpus / SciFact / TREC-COVID / TREC-News (macro)																					BM25: 0.416 0.693			
Pre-retrieval	Query2doc	0.456	0.765	0.455	0.767	0.464	0.767	0.456	0.768	0.462	0.763	0.453	0.759	0.460	0.768	0.457	0.761	0.472	0.773	0.463	0.763			
	MuGI	0.457	0.767	0.461	0.773	0.465	0.774	0.461	0.772	0.469	0.774	0.447	0.761	0.463	0.772	0.457	0.771	0.464	0.778	0.454	0.771			
Post-retrieval	LameR	0.478	0.771	0.485	0.763	0.485	0.771	0.479	0.769	0.491	0.763	0.462	0.742	0.476	0.742	0.487	0.771	0.504	0.780	0.485	0.765			
	CSQE	0.472	0.759	0.490	0.768	0.490	0.771	0.485	0.766	0.501	0.769	0.488	0.766	0.496	0.772	0.489	0.767	0.500	0.779	0.484	0.765			
Iterative	ThinkQE	0.475	0.777	0.479	0.775	0.483	0.776	0.483	0.779	0.487	0.778	0.475	0.770	0.490	0.775	0.494	0.778	0.499	0.785	0.487	0.781			
	ADORE	0.482	0.779	0.493	0.783	0.488	0.781	0.497	0.781	0.498	0.786	0.486	0.766	0.485	0.760	0.494	0.764	0.516	0.780	0.495	0.765			
BRIGHT AoPS / Bio. / Earth / Econ. / LeetCode / Pony / Psych. / Robot. / StackOv. / Sust. / TQA-Q / TQA-T (macro)																					BM25: 0.137 0.634			
Pre-retrieval	Query2doc	0.182	0.702	0.182	0.697	0.182	0.703	0.181	0.703	0.189	0.708	0.181	0.700	0.184	0.704	0.173	0.693	0.189	0.706	0.182	0.698			
	MuGI	0.229	0.758	0.238	0.756	0.252	0.765	0.249	0.762	0.271	0.783	0.274	0.785	0.296	0.798	0.259	0.762	0.272	0.770	0.271	0.763			
Post-retrieval	LameR	0.195	0.719	0.222	0.747	0.226	0.747	0.220	0.735	0.235	0.756	0.225	0.747	0.221	0.741	0.223	0.743	0.254	0.765	0.248	0.769			
	CSQE	0.214	0.751	0.227	0.765	0.235	0.760	0.231	0.763	0.255	0.775	0.242	0.770	0.262	0.775	0.214	0.748	0.235	0.763	0.223	0.759			
Iterative	ThinkQE	0.253	0.766	0.257	0.765	0.260	0.770	0.265	0.768	0.282	0.777	0.265	0.771	0.277	0.787	0.255	0.749	0.270	0.756	0.267	0.762			
	ADORE	0.247	0.764	0.264	0.758	0.277	0.759	0.287	0.760	0.305	0.787	0.316	0.766	0.304	0.757	0.287	0.752	0.307	0.757	0.290	0.749			

improvements and three declines, and ReFormer [5] records 12 improvements and three declines.

We measure generation variability using three runs for each of 468 method-collection-configuration combinations across the nine Qwen-family models and four DL collections. These runs use different random seeds. Across these combinations, the median standard deviation of nDCG@10 is 0.0082, with a maximum of 0.0417. At Qwen2.5-7B, the median is 0.0068. Using this LLM, ADORE [2], ThinkQE [27], LameR [41], CSQE [26], and MuGI [51] achieve gains over BM25 of at least nine times the corresponding standard deviation on each TREC DL collection.

Higher nDCG@10 does not always mean higher recall. Every expansion configuration in Table 3 improves average Recall@1000 over BM25 on TREC DL, BEIR [45], and BRIGHT [43]. On RAGgy-dev [34], all displayed configurations improve nDCG@10 but reduce Recall@1000. Recall ranges from 0.501 to 0.570, compared with 0.574 for BM25. On this collection, these configurations improve ranking quality within the top ten while retrieving a smaller fraction of judged relevant documents within the top 1,000.

Incomplete relevance judgments also limit the evaluation [6]. On DL22 and DL23 [8, 10], the leading methods with GPT-5.5 retrieve

18–20% unjudged documents in their top ten, while BM25’s top ten are fully judged. Unjudged documents are treated as non-relevant, but their relevance remains unknown. Further judgments would be needed to determine their effect on the comparison with BM25.

6.2 Expansion with Small LLMs

Our experiments show that query expansion methods get less reliable with a smaller LLM. Using Qwen2.5-0.5B as our backbone, 62.8% of the 78 method-collection combinations do not improve over BM25. Mean nDCG@10 across the 13 methods and six primary collections is 0.4249, compared with 0.4365 for BM25. Once we switch to Qwen2.5-72B as the backbone, the proportion of combinations that do not improve over BM25 falls to 3.8%.

Four methods improve mean nDCG@10 across the six primary collections even at 0.5B. ADORE [2] gains 0.042, LameR [41] 0.039, ThinkQE [27] 0.036, and MuGI [51] 0.019. On the other hand, Q2E [20], GenQREnsemble [11], and GenQR [47] have the largest average declines, at 0.068, 0.049, and 0.043. We cannot attribute this pattern to corpus feedback. For instance, MuGI [51] improves over BM25 without retrieved context, while CSQE [26] and ReFormer [5] use retrieved context but fall below BM25.

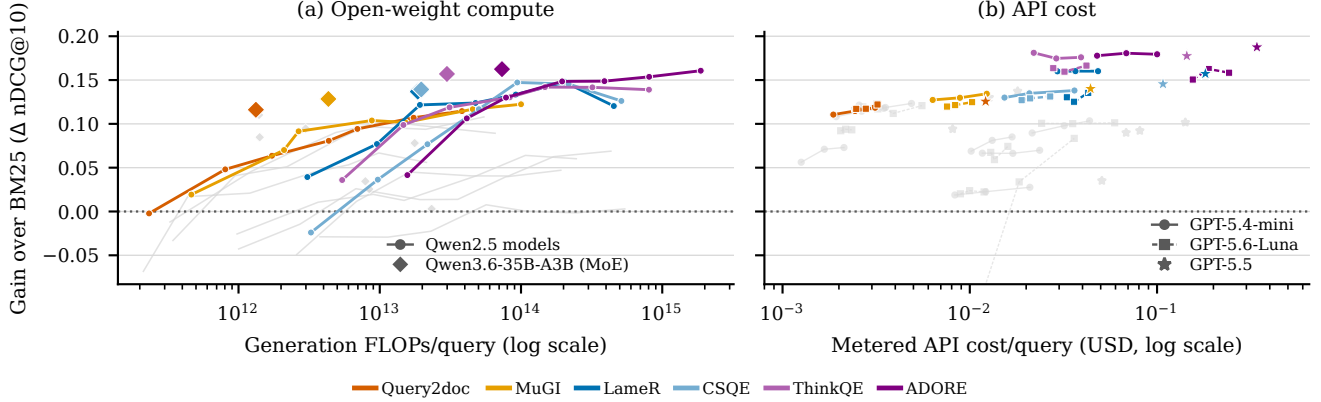


Figure 2: Expansion effectiveness and generation cost. Mean improvement in nDCG@10 over BM25 is averaged across the six primary collections. The horizontal axes show estimated generation FLOPs per query for open-weight models in panel (a) and metered API cost per query in panel (b), both on a logarithmic scale.

Backbone choice also changes the relative ordering of expansion methods. Ranking the 13 methods by mean nDCG@10 across the six primary collections results in a Kendall’s τ [23] of only 0.62 between Qwen2.5-0.5B and Qwen2.5-72B, and 0.59 between Qwen2.5-0.5B and GPT-5.5. We expect $\tau = 1$ if two rankings are identical.

Moreover, our results at Qwen2.5-0.5B vary more across random seeds. On the four DL collections, the median standard deviation of nDCG@10 is 0.0127 at Qwen2.5-0.5B, compared with 0.0068 at Qwen2.5-7B. We also compare the gains over BM25 with the standard deviation across runs. In 29 of 52 method–collection combinations at Qwen2.5-0.5B, the gain is less than three times the corresponding standard deviation. The corresponding counts are three at Qwen2.5-14B and one at Qwen2.5-72B.

Individual examples show both topic drift and failures to follow the requested format. For the DL19 query *types of dysarthria from cerebral palsy*, BM25 scores 0.560 nDCG@10. Using Qwen2.5-0.5B, Q2E generates terms about aphasia and scores 0.121. LameR begins with expansion text but switches to a multiple-choice quiz and scores 0.233. In both cases, the generated text reduces retrieval effectiveness on a query that BM25 already handles well.

6.3 Ablation Study

The methods combine several choices, including the format of the expansion text, retrieved context [26, 41], multiple generated expansions [11, 51], and repeated generation and retrieval [2, 27]. We study each choice using controlled ablations with shared prompt templates and rules for combining the query and expansion.

Table 4 compares three formats for query expansion. Keyword generation asks the model to list terms related to the query, passage generation asks it to write a passage relevant to the query, and answer generation asks it to answer the query directly. In all three cases, we use the generated text to expand the query. We evaluate each format with and without the top five BM25 passages as context, using three random seeds on DL19, DL20, DL22, and DL23. Adding retrieved context improves mean nDCG@10 by 0.113 at 0.5B and 0.091 at 7B, averaged over the three formats. Passage generation outperforms keyword generation across the tested settings. At 0.5B,

Table 4: Adding retrieved context improves mean nDCG@10 for all three output formats at both model sizes. Cells report mean nDCG@10 over DL19/20/22/23 and three seeds. Bold scores exceed the BM25 baseline at 0.379.

Qwen2.5	retrieved context	what the model is asked to generate		
		keywords	passage	answer
0.5B	none	0.194	0.295	0.266
	top-5 BM25	0.321	0.384	0.390
7B	none	0.303	0.413	0.417
	top-5 BM25	0.429	0.493	0.485

only passage and answer generation with retrieved context improve over BM25. At 7B, both passage and answer generation formats improve over BM25 even without retrieved context.

Figure 3 shows an ablation on DL19/20/22/23 using Qwen2.5 at 0.5B, 3B, 14B, and 32B. Scores are averaged over the four collections and three seeds. We start from one expansion, add retrieved context, and then combine five expansions, comparing one versus three rounds. In the one-round comparisons, adding retrieved context to one expansion improves mean nDCG@10 by 0.0758, 0.0938, 0.0771, and 0.0546 at 0.5B, 3B, 14B, and 32B, respectively. Going from one expansion to five with retrieved context improves mean nDCG@10 further, by 0.0444, 0.0085, 0.0099, and 0.0078, respectively. Additional rounds do not consistently improve effectiveness. The strong performance of iterative methods therefore does not imply that adding more rounds consistently improves a given method.

Generation and extraction. We also compare generated expansions with evidence extracted from the same top ten BM25 passages. Two controls append either RM3-weighted terms [1] or selected feedback sentences to the query. Both are limited to LameR’s mean output length. Across the nine collections in this comparison, sentence extraction reduces mean nDCG@10 by 0.064 relative to BM25, and term extraction reduces it by 0.102. LameR and CSQE improve

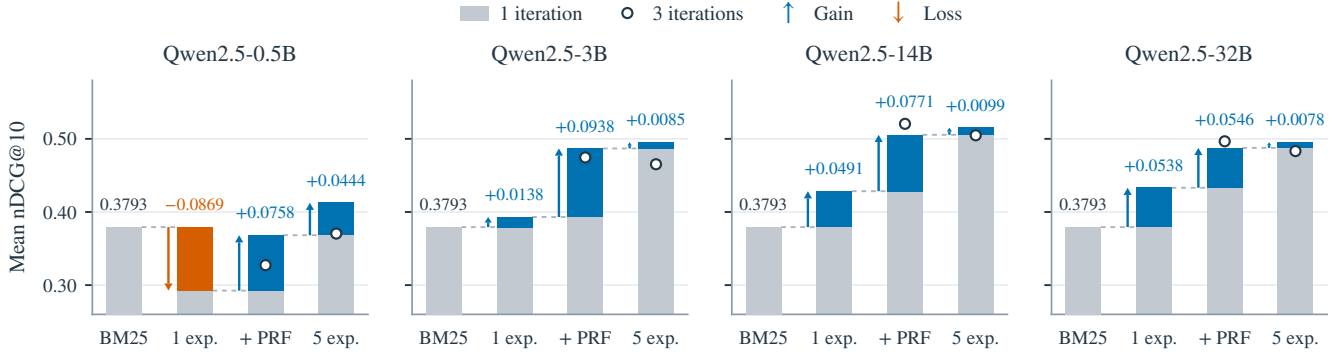


Figure 3: Adding retrieved context (PRF) gives the largest incremental gain at all four model sizes in the one-round comparisons, while combining five expansions improves mean nDCG@10 further. Scores are averaged over DL19/20/22/23 and three seeds. Bars show one round and hollow circles three. Signed labels show changes from the preceding setting.

over BM25 by 0.092 and 0.074, respectively. These results favor generated expansions over the two extraction methods tested here.

6.4 Model Size and Family

Increasing Qwen2.5 from 0.5B to 32B raises mean nDCG@10 from 0.4249 to 0.5315 across the 13 methods and six primary collections, with 143× the estimated generation FLOPs. Increasing from 32B to 72B adds another 2.4× in compute but produces a slightly lower score of 0.5300, as shown in Figure 2(a).

For Qwen2.5-32B and 72B, we average three runs for each method and query on all six primary collections, then apply a paired bootstrap [14] with 10,000 resamples. The mean per-query difference between 72B and 32B across methods is -0.0015 nDCG@10, with a 90% confidence interval of $[-0.0021, +0.0018]$. The entire interval lies within our ± 0.005 equivalence margin [25], supporting equivalent average effectiveness. Individual methods still differ. For example, LameR falls from 0.584 at 32B to 0.557 at 72B. Gemma-4-31B reaches a mean nDCG@10 of 0.5284, close to Qwen2.5-32B and 72B. However, larger Gemma-4 models do not consistently improve retrieval effectiveness. Across E4B, 12B, and 31B, LameR scores 0.523, 0.500, and 0.588, while ADORE scores 0.537, 0.606, and 0.591.

Qwen3.6-35B-A3B, which is a mixture-of-experts model with 35B total and 3B active parameters, reaches 0.5342 while using about 1/11th of Qwen2.5-32B’s estimated generation FLOPs. With Query2doc [46], it improves the primary-collection mean from 0.517 with Qwen2.5-7B to 0.553 at roughly one-third of the generation FLOPs. Qwen3.5-9B with thinking disabled also improves on Qwen2.5-7B, by 0.023 on BRIGHT and 0.003 on the six primary collections. These comparisons show that model size alone does not determine expansion effectiveness or generation cost.

7 RQ3: Cost and Effectiveness

Table 5 reports each method’s mean nDCG@10 and mean estimated generation FLOPs per query across nine Qwen models and nine collections. These collections comprise the six primary collections, DL-Hard, TREC-COVID, and FiQA. Mean generation compute varies by 64× across methods. Among the 12 methods with positive mean

Table 5: Expansion effectiveness and generation compute. Methods are ranked by mean nDCG@10 across nine Qwen configurations on DL19/20/22/23, RAGgy-dev, SciFact, DL-Hard, TREC-COVID, and FiQA. † shows Pareto frontier of (FLOPs/q, Δ). +/− Holm-corrected wins/losses vs. BM25 across the 78 primary collection-configuration pairs.

Method	Mean	+/−	Δ	Gen. FLOPs/q	min Δ/σ
ADORE†	0.5331	67/0	+0.1183	3.7×10^{14}	15.5
ThinkQE†	0.5241	74/1	+0.1093	1.7×10^{14}	9.4
LameR†	0.5091	72/0	+0.0943	1.2×10^{14}	19.1
CSQE†	0.4995	64/3	+0.0847	1.1×10^{14}	14.9
MuGI†	0.4933	54/1	+0.0785	2.3×10^{13}	12.2
Query2doc†	0.4787	57/1	+0.0639	8.8×10^{12}	5.1
HyDE	0.4708	55/1	+0.0560	1.2×10^{13}	3.8
CoT	0.4693	56/1	+0.0545	1.4×10^{13}	7.0
Q2E†	0.4531	44/3	+0.0383	5.8×10^{12}	4.0
GenQREnsemble	0.4485	38/1	+0.0337	9.1×10^{13}	3.6
AMD	0.4291	27/3	+0.0143	5.3×10^{13}	0.3
GenQR	0.4275	20/2	+0.0127	4.1×10^{13}	0.3
ReFormerR	0.4118	12/3	−0.0030	1.1×10^{14}	−3.1
BM25 (no expansion)	0.4148	—	—	0	—

gains over BM25, the ratio of mean gain to mean generation compute varies by 27×. ADORE [2] achieves the largest mean gain, 0.118 nDCG@10, at 3.7×10^{14} FLOPs per query. MuGI [51] achieves 66% of this gain at 1/16 of ADORE’s compute, and Query2doc [46] achieves 54% at 1/42. ReFormerR [5] uses about one-third of ADORE’s compute but scores 0.003 below BM25 on average. These FLOP estimates exclude retrieval computation and do not measure response time.

Based on these averages, seven of the 13 expansion methods lie on the Pareto frontier. A method is on this frontier if no other expansion method is at least as effective and uses no more generation compute, with a strict improvement in either measure. The frontier extends from Q2E at the lowest cost to ADORE at the highest effectiveness. HyDE, CoT, GenQR, AMD, GenQREnsemble, and ReFormerR each have a cheaper alternative with higher mean

nDCG@10. These comparisons depend on the evaluated collections and settings. GenQREnsemble, for example, leads on individual BRIGHT subsets despite its lower mean here (§8).

We also examine generation budgets and query construction on DL19, DL20, DL22, and DL23, averaging nDCG@10 over three seeds and the four collections. For the budget comparisons, we count all generated completions, including those used for ADORE’s relevance judgments. With Qwen2.5-7B, increasing MuGI from five to eight generations and adding a third round to ThinkQE each yield less than 0.003 nDCG@10 per additional generated completion. Increasing ADORE’s maximum number of rounds from three to five improves mean nDCG@10 by only 0.0008 in total.

Query construction also affects effectiveness. We test Query2doc with one, three, five, and ten repetitions of the original query before appending the generated text. At Qwen2.5-0.5B, the difference between the highest and lowest mean nDCG@10 across these settings is 0.137. Comparisons of generation cost and effectiveness therefore need to specify both the generation budget and how the generated text is combined with the original query.

API cost and reasoning budget. In Figure 2(b), we compare mean nDCG@10 gains over BM25 with metered API cost per query on the six primary collections. GPT-5.4-mini at low reasoning effort reaches a mean nDCG@10 of 0.5454 across methods. Its gain over BM25 is 87.7% of GPT-5.5’s gain at low reasoning. The corresponding total API spending on these collections is \$122 and \$818, respectively, so GPT-5.4-mini costs 14.9% as much. Increasing GPT-5.4-mini’s reasoning budget from low to high adds 0.007 to the overall mean nDCG@10, while total output tokens increase from 13.0M to 39.2M and total spending rises to \$240.

The effect of the reasoning budget varies by method. With GPT-5.4-mini at low, medium, and high reasoning, ADORE scores 0.614, 0.617, and 0.616. Its API cost rises from \$0.048 per query at low reasoning to \$0.100 at high reasoning. ThinkQE scores 0.618, 0.611, and 0.612 in the same order, while its cost rises from \$0.022 to \$0.039 per query between low and high reasoning.

GPT-5.6-Luna [33] gains 0.021 in mean nDCG@10 across methods when the reasoning budget increases from low to high. The changes again vary by method. At low, medium, and high reasoning, ThinkQE scores 0.600, 0.596, and 0.603, and ADORE scores 0.587, 0.599, and 0.595. At high reasoning, Luna’s overall mean is 0.5481, compared with 0.5607 for GPT-5.5 at low reasoning. Higher reasoning budgets therefore need to be evaluated for each method. Among the configurations tested here, GPT-5.4-mini at low reasoning provides much of GPT-5.5’s improvement over BM25 at substantially lower API cost.

8 RQ4: Effectiveness Across Collections

Figure 4 compares method rankings across 25 collections using Qwen3.5-9B. Post-retrieval methods lead on 22 collections, while iterative methods account for 21 of these wins. Pre-retrieval methods lead on three collections, all of which are BRIGHT subsets.

For each tested open-weight configuration, pre-retrieval methods lead on three to seven of the 12 BRIGHT subsets [43]. GenQR [47] and GenQREnsemble [11], which rank near the bottom on the primary collections, lead on these subsets, including TheoremQA-Theorems, Biology, and LeetCode.

On BRIGHT, we select the best overall method, which achieves the highest mean nDCG@10 across 12 subsets, and the oracle, which is the highest-scoring method for each subset. The oracle gives the best scores among the tested methods and sets the upper bound. We find that the best overall method captures most of the oracle’s improvement over BM25, although which method is best depends on the backbone. For instance, MuGI [51] achieves 96% of this gain with Qwen2.5-0.5B, ThinkQE [27] achieves 91% with Qwen2.5-7B, and ADORE [2] achieves 92% with Qwen3.5-9B.

Across the full set of 25 collections, expansion does not always improve over BM25. With Qwen3.5-9B, ADORE achieves the highest mean nDCG@10 but falls below BM25 on ArguAna. Query expansion can also reduce effectiveness on FiQA with small LLMs. The benefit of expansion therefore needs to be checked on the target collection, even for methods that perform well on average.

9 Conclusion

Published gains in LLM-based query expansion are difficult to compare because studies use different LLMs, retrieval systems, and evaluation settings. We present Expandorini, a framework for reproducible, cost-aware evaluation of LLM-based query expansion, and use it to benchmark 13 methods across 22 LLM configurations and 25 collections under controlled BM25 retrieval settings.

Our implementations improve over BM25 in most TREC DL comparisons, except ReFormeR on DL19, consistent with the gains reported in the original studies. Larger models do not always improve effectiveness, and we observe no meaningful average improvement from Qwen2.5-32B to 72B across the six primary collections. Controlled ablations show benefits from combining multiple expansions and generating passages or answers instead of keywords. Retrieved context improves mean effectiveness in the DL ablations, while additional rounds provide limited average improvements in the tested settings. ADORE achieves the highest average effectiveness in our cost comparison, while MuGI achieves 66% of ADORE’s mean gain over BM25 at 1/16 of its estimated generation compute. Higher reasoning budgets do not consistently improve individual methods. Post-retrieval and iterative methods lead on most collections.

Our evaluation uses English queries and BM25 retrieval, and the generation cost estimates exclude retrieval computation. Differences in LLMs and implementations limit direct comparisons with published results. We keep each method’s prompt template fixed across LLMs, so our results do not show whether tuning the prompts for each model would change the rankings. The API experiments cover one provider, and possible overlap between benchmark data and LLM training data remains unresolved. Extending the evaluation to dense, hybrid, and multilingual retrieval would test how broadly the findings apply.

Acknowledgments

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

		TREC DL / DL-Hard					RAG	BEIR							BRIGHT													
		DL19	DL20	DL22	DL23	DL-Hard	RAGgy-dev	ArguAna	DBpedia	FiQA	NFCorpus	SciFact	TREC-COVID	TREC-News	AopS	Biology	Earth Science	Economics	LeetCode	Pony	Psychology	Robotics	StackOverflow	Sustainable Living	TheoremQA-Q	TheoremQA-T		
Query2doc	Q2E	7	7	9	9	5	5	2	8	4	4	8	5	6	6	11	11	12	7	11	12	10	12	12	12	11		
	CoT	8	9	6	8	10	12	4	9	10	11	9	7	11	4	7	7	8	10	6	8	9	8	7	8	8		
	HyDE	5	6	5	6	8	8	6	6	8	10	6	9	8	7	10	10	10	11	9	11	11	9	11	10	10		
	AMD	6	5	7	4	7	9	7	5	9	9	7	8	10	2	9	6	9	7	8	9	8	10	9	9	12		
	GenQR	12	13	13	12	12	13	8	13	13	13	13	13	13	9	12	12	11	6	12	10	13	11	10	11	9		
GenQREnsemble	GenQR	11	11	11	10	13	11	10	11	12	6	12	11	7	5	3	5	4	1	5	5	5	4	4	2	2		
	MuGI	10	8	10	11	9	10	11	10	11	3	10	12	4	13	1	3	2	2	2	4	4	2	2	3	1		
	LameR	2	4	3	5	3	7	9	3	7	7	5	6	9	3	4	2	6	12	4	2	3	3	5	6	6		
	CSQE	9	10	8	7	5	2	3	7	5	8	3	2	5	11	8	8	7	4	10	7	7	7	8	7	4		
	ReFormer	3	2	4	3	2	3	1	4	2	2	2	3	3	8	6	9	3	9	3	6	6	6	6	4	7		
ADORE	ThinkQE	13	12	12	13	11	6	5	12	6	12	11	10	12	12	13	13	13	5	13	13	12	13	13	13	13		
	ThinkQE	3	3	2	2	4	1	12	2	3	5	4	1	1	1	2	1	5	13	7	3	2	1	3	5	5		
	ADORE	1	1	1	1	1	4	13	1	1	1	1	4	2	10	5	4	1	3	1	1	1	5	1	1	3		

Figure 4: Within-collection nDCG@10 rankings of 13 query expansion methods across 25 collections using Qwen3.5-9B. Post-retrieval methods lead on 22 collections, with iterative methods accounting for 21 of these wins.

References

- # References
- [1] Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004 (NIST Special Publication, Vol. 500-261)*, Ellen M. Voorhees and Lori P. Buckland (Eds.). National Institute of Standards and Technology (NIST). <http://trec.nist.gov/pubs/trec13/papers/umass.novelty.hard.pdf>
 - [2] Amin Bigdeli, Negar Arabzadeh, Radin Hamidi Rad, Sajad Ebrahimi, Charles L. A. Clarke, and Ebrahim Bagheri. 2026. ADORÉ: Iterative Query Expansion with Retrieval-Grounded Relevance Feedback. arXiv:2606.13905 [cs.IR] <https://arxiv.org/abs/2606.13905>
 - [3] Amin Bigdeli, Radin Hamidi Rad, Mert Incesu, Negar Arabzadeh, Charles Clarke, and Ebrahim Bagheri. 2026. QueryGym: A Toolkit for Reproducible LLM-Based Query Reformulation. In *Companion Proceedings of the ACM Web Conference 2026 (Dubai, United Arab Emirates) (WWW Companion '26)*. Association for Computing Machinery, New York, NY, USA, 196–199. doi:10.1145/3774905.3793135
 - [4] Amin Bigdeli, Radin Hamidi Rad, Hai Son Le, Mert Incesu, Negar Arabzadeh, Charles L. A. Clarke, and Ebrahim Bagheri. 2026. A Reproducibility Study of LLM-Based Query Reformulation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (Melbourne, VIC, Australia) (SIGIR '26)*. Association for Computing Machinery, New York, NY, USA, 2824–2834. doi:10.1145/3805712.3808560
 - [5] Amin Bigdeli, Mert Incesu, Negar Arabzadeh, Charles L. A. Clarke, and Ebrahim Bagheri. 2026. ReFormerR: Learning and Applying Explicit Query Reformulation Patterns. In *Advances in Information Retrieval*, Ricardo Campos, Adam Jatowt, Yanyan Lan, Mohammad Aliannejadi, Christine Bauer, Sean MacAvaney, Avishek Anand, Zhaochun Ren, Suzan Verberne, Nan Bai, and Masoud Mansoury (Eds.). Springer Nature Switzerland, Cham, 400–408.
 - [6] Chris Buckley and Ellen M. Voorhees. 2004. Retrieval Evaluation with Incomplete Information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (Sheffield, United Kingdom) (SIGIR '04)*. Association for Computing Machinery, New York, NY, USA, 25–32. doi:10.1145/1008992.1009000
 - [7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 Deep Learning Track. In *Text REtrieval Conference (TREC)*. TREC. <https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2020-deep-learning-track/>
 - [8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2023. Overview of the TREC 2022 Deep Learning Track. In *Text REtrieval Conference (TREC)*. NIST, TREC. <https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2022-deep-learning-track/>
 - [9] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 Deep Learning track. In *Text REtrieval Conference (TREC)*. TREC. <https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2019-deep-learning-track/>
 - [10] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Hossein A. Rahmani, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2024. Overview of the TREC 2023 Deep Learning Track. In *Text REtrieval Conference (TREC)*. NIST, TREC. <https://www.microsoft.com/en-us/research/publication/overview-of-the-trec-2023-deep-learning-track/>
 - [11] Kaustubh D. Dhole and Eugene Agichtein. 2024. GenQREnsemble: Zero-Shot LLM Ensemble Prompting for Generative Query Reformulation. In *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part III*. Springer-Verlag, Berlin, Heidelberg, 326–335. doi:10.1007/978-3-031-56063-7_24
 - [12] Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show Your Work: Improved Reporting of Experimental Results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 2185–2194. doi:10.18653/v1/D19-1224
 - [13] Earendil Works. 2026. pi: AI Agent Toolkit—Unified LLM API, Agent Loop, and Coding-Agent CLI. <https://github.com/earendil-works/pi>
 - [14] B. Efron. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* 7, 1 (1979), 1–26. <https://www.jstor.org/stable/2958830>
 - [15] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (Virtual Event, Canada) (SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2288–2292. doi:10.1145/3404835.3463098
 - [16] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The Vocabulary Problem in Human-System Communication. *Commun. ACM* 30, 11 (Nov. 1987), 964–971. doi:10.1145/32206.32212
 - [17] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1762–1777. doi:10.18653/v1/2023.acl-long.99
 - [18] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70. <http://www.jstor.org/stable/4615733>
 - [19] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. arXiv:2112.09118 [cs.LG]

- <https://arxiv.org/abs/2112.09118>
- [20] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query Expansion by Prompting Large Language Models. arXiv:2305.03653 [cs.IR] <https://arxiv.org/abs/2305.03653>
 - [21] Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy Lin. 2020. Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II*. Springer-Verlag, Berlin, Heidelberg, 28–34. doi:10.1007/978-3-030-45442-5_4
 - [22] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG] <https://arxiv.org/abs/2001.08361>
 - [23] M. G. Kendall. 1938. A New Measure of Rank Correlation. *Biometrika* 30, 1/2 (1938), 81–93. <http://www.jstor.org/stable/2332226>
 - [24] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP 2023)*, 611–626. doi:10.1145/3600006.3613165
 - [25] Daniel Lakens. 2017. Equivalence Tests: A Practical Primer for t Tests, Correlations, and Meta-Analyses. *Social Psychological and Personality Science* 8, 4 (2017), 355–362. PMID: 28736600. doi:10.1177/1948550617697177
 - [26] Yibin Lei, Yu Cao, Tianyi Zhou, Tao Shen, and Andrew Yates. 2024. Corpus-Steered Query Expansion with Large Language Models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, Yvette Graham and Matthew Purver (Eds.). Association for Computational Linguistics, St. Julian's, Malta, 393–401. doi:10.18653/v1/2024.eacl-short.34
 - [27] Yibin Lei, Tao Shen, and Andrew Yates. 2025. ThinkQE: Query Expansion via an Evolving Thinking Process. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 17772–17781. doi:10.18653/v1/2025.findings-emnlp.965
 - [28] Jimmy Lin. 2022. Building a Culture of Reproducibility in Academic Research. arXiv:2212.13534 [cs.IR] <https://arxiv.org/abs/2212.13534>
 - [29] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (Virtual Event, Canada) (SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2356–2362. doi:10.1145/3404835.3463238
 - [30] Iain Mackie, Jeffrey Dalton, and Andrew Yates. 2021. How Deep Is Your Learning: The DL-Hard Annotated Deep Learning Dataset. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (Virtual Event, Canada) (SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2335–2341. doi:10.1145/3404835.3463262
 - [31] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. (November 2016). <https://www.microsoft.com/en-us/research/publication/ms-marco-human-generated-machine-reading-comprehension-dataset/>
 - [32] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. arXiv:1904.08375 [cs.IR] <https://arxiv.org/abs/1904.08375>
 - [33] OpenAI. 2026. GPT-5.6 System Card. <https://deploymentsafety.openai.com/gpt-5-6/>
 - [34] Ronak Pradeep, Nandan Thakur, Sahel Sharifmoghammad, Eric Zhang, Ryan Nguyen, Daniel Campos, Nick Craswell, and Jimmy Lin. 2025. Ragnarök: A Reusable RAG Framework and Baselines for TREC 2024 Retrieval-Augmented Generation Track. In *Advances in Information Retrieval*, Claudia Hauff, Craig Macdonald, Dietmar Jannach, Gabriella Kazai, Franco Maria Nardini, Fabio Pinelli, Fabrizio Silvestri, and Nicola Tonello (Eds.). Springer Nature Switzerland, Cham, 132–148.
 - [35] Qwen Team. 2026. Qwen3.5: Towards Native Multimodal Agents. <https://huggingface.co/Qwen/Qwen3.5-9B>
 - [36] Qwen Team. 2026. Qwen3.6-35B-A3B Model Card. <https://huggingface.co/Qwen/Qwen3.6-35B-A3B>
 - [37] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (April 2009), 333–389. doi:10.1561/15000000019
 - [38] J. J. Rocchio. 1971. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, Gerard Salton (Ed.). Prentice-Hall, Englewood Cliffs, NJ, 313–323.
 - [39] Harrison Scells, Shengyao Zhuang, and Guido Zucco. 2022. Reduce, Reuse, Recycle: Green Information Retrieval Research. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (Madrid, Spain) (SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2825–2837. doi:10.1145/3477495.3531766
 - [40] Wonduk Seo, Hyunjin An, and Seunghyun Lee. 2026. A New Query Expansion Approach for Enhancing Information Retrieval via Agent-Mediated Dialogic Inquiry. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining (Boise, ID, USA) (WSDM '26)*. Association for Computing Machinery, New York, NY, USA, 1232–1237. doi:10.1145/3773966.3779360
 - [41] Tao Shen, Guodong Long, Xiubo Geng, Chongyang Tao, Yibin Lei, Tianyi Zhou, Michael Blumenstein, and Daxin Jiang. 2024. Retrieval-Augmented Retrieval: Large Language Models are Strong Zero-Shot Retriever. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 15933–15946. doi:10.18653/v1/2024.findings-acl.943
 - [42] Mark D. Smucker, James Allan, and Ben Carterette. 2007. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (Lisbon, Portugal) (CIKM '07)*. Association for Computing Machinery, New York, NY, USA, 623–632. doi:10.1145/1321440.1321528
 - [43] 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, Seran O Arik, Danqi Chen, and Tao Yu. 2025. BRIGHT: A Realistic and Challenging Benchmark for Reasoning-Intensive Retrieval. In *The Thirteenth International Conference on Learning Representations*. <https://openreview.net/forum?id=ykuc5q381b>
 - [44] Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuyang Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] <https://arxiv.org/abs/2412.15115>
 - [45] 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)*. <https://openreview.net/forum?id=wCu6T5xFeJ>
 - [46] Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query Expansion with Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 9414–9423. doi:10.18653/v1/2023.emnlp-main.585
 - [47] Xiao Wang, Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2023. Generative Query Reformulation for Effective Adhoc Search. arXiv:2308.00415 [cs.IR] <https://arxiv.org/abs/2308.00415>
 - [48] Orion Weller, Kyle Lo, David Wadden, Dawn Lawrie, Benjamin Van Durme, Arman Cohan, and Luca Soldaini. 2024. When do Generative Query and Document Expansions Fail? A Comprehensive Study Across Methods, Retrievers, and Datasets. In *Findings of the Association for Computational Linguistics: EACL 2024*, Yvette Graham and Matthew Purver (Eds.). Association for Computational Linguistics, St. Julian's, Malta, 1987–2003. doi:10.18653/v1/2024.findings-eacl.134
 - [49] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (Shinjuku, Tokyo, Japan) (SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1253–1256. doi:10.1145/3077136.3080721
 - [50] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained Transformers for Text Ranking: BERT and Beyond. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorials*, Greg Kondrak, Kalina Bontcheva, and Dan Gillick (Eds.). Association for Computational Linguistics, Online, 1–4. doi:10.18653/v1/2021.naacl-tutorials.1
 - [51] Le Zhang, Yihong Wu, Qian Yang, and Jian-Yun Nie. 2024. Exploring the Best Practices of Query Expansion with Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 1872–1883. doi:10.18653/v1/2024.findings-emnlp.103