MMLF: Multi-query Multi-passage Late Fusion Retrieval

Yuan-Ching Kuo[1], Yi Yu[1,2], Chih-Ming Chen[1], Chuan-Ju Wang[1]
[1]Academia Sinica, [2]The Ohio State University
{yckuo,cmchen,cjwang}@citi.sinica.edu.tw, yu.4063@osu.edu

Abstract

Leveraging large language models (LLMs) for query expansion has proven highly effective across diverse tasks and languages. Yet, challenges remain in optimizing query formatting and prompting, often with less focus on handling retrieval results. In this paper, we introduce Multi-query Multi-passage Late Fusion (MMLF), a straightforward yet potent pipeline that generates sub-queries, expands them into pseudo-documents, retrieves them individually, and aggregates results using reciprocal rank fusion. Our experiments demonstrate that MMLF exhibits superior performance across five BEIR benchmark datasets, achieving an average improvement of 4% and a maximum gain of up to 8% in both Recall@1k and nDCG@10 compared to state of the art across BEIR information retrieval datasets.[1]

1 Introduction

Information retrieval (IR) aims to identify relevant documents from large corpora in response to user queries. Despite extensive research, ad hoc retrieval continues to pose challenges, particularly with brief or ambiguous queries that complicate accurate inference of user intent.

Traditional IR systems, such as BM25, rely on exact term matching, which often struggles with limited lexical overlap between queries and documents. To address this, query expansion techniques use lexical knowledge bases, leveraging resources like ontologies or semantic networks to add semantically related terms, including synonyms (Bhogal et al., 2007; Qiu and Frei, 1993; Voorhees, 1994). With the advent of dense retrieval models, query reformulation has shifted from lexical matching to semantic similarity. These models embed queries and documents into shared vector spaces, facilitating semantic matching even when lexical overlap is minimal (Lee et al., 2019; Karpukhin et al., 2020).

Recently, large language models (LLMs) have shown remarkable capabilities in natural language understanding and generation (Brown et al., 2020; Chen et al., 2021). Serving as external knowledge bases, LLMs apply their learned knowledge to improve sparse and dense retrieval systems through query reformulation. For instance, while Query2Doc expands a query into a passage to better align with document content (Wang et al., 2023), Chain-of-Thought (CoT) enriches the query with a step-by-step rationale to better meet complex information needs (Jagerman et al., 2023). Later, MILL introduces the query-query-document (QQD) pipeline, which segments the query into sub-queries and generates corresponding passages in a single step, using a single prompt for the LLM. This approach produces diverse, contextual information that reflects the underlying search intent, particularly in dense retrieval contexts (Jia et al., 2024). However, the QQD pipeline concatenates these generated passages into a single text for retrieval, potentially blending multiple perspectives and diluting their distinct contributions.

To address these challenges, we propose MMLF, a two-stage pipeline that separates sub-query generation from passage expansion. Subsequently, we perform independent retrieval for each generated passage and the original query, aggregating the results using reciprocal rank fusion (RRF). Unlike RAG-Fusion (Rackauckas, 2024), which is tailored for chatbot applications and retrieves documents directly without passage expansion, MMLF focuses on enhancing broader information retrieval tasks through a two-step pipeline that generates diverse and contextually enriched passages for retrieval. The effectiveness of the proposed pipeline is demonstrated across five BEIR benchmark datasets (Thakur et al., 2021), achieving an average improvement of 4% and a maximum gain

[1]Codes and generated artifacts are available at: https://anonymous.4open.science/r/MMLF/

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 6602–6613

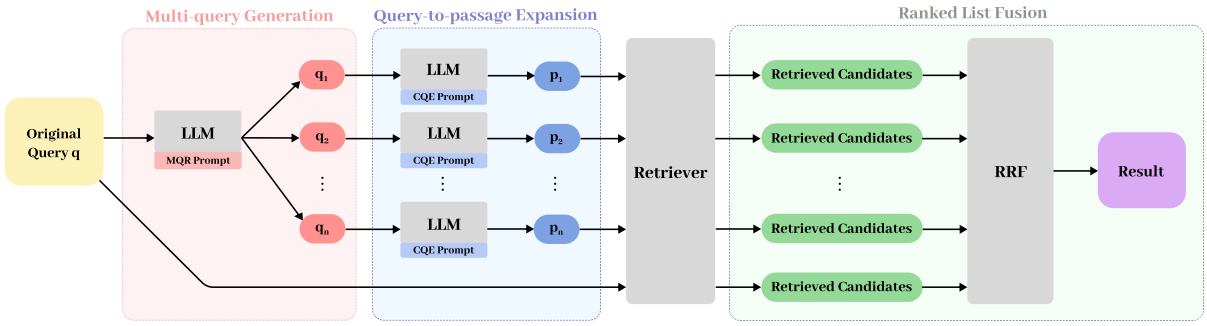

Figure 1: The MMLF pipeline

of up to 8% in both Recall@1k and nDCG@10 compared to the state-of-the-art approach, MILL. In addition, we conduct comprehensive ablation studies to evaluate different variants of MMLF, including various fusion methods, the role of the original query, and alternative query reformation approaches. These studies provide valuable insight into the rationale behind our design and further confirm the robustness of the proposed MMLF.

2 The Proposed Pipeline: MMLF

In this section, we introduce the MMLF pipeline, which enhances retrieval through three steps: generating diverse sub-queries, expanding them into contextual passages, and fusing rankings via reciprocal rank fusion (RRF).

The first two stages of MMLF are central to the query reformulation process. They leverage the advantages of multi-query generation, capturing various interpretations of user intent and query-to-passage expansion, enriching these interpretations with detailed context. Unlike the concatenation method used by our main competitor, MILL, late-fusion techniques like RRF maximize the utility of every individual sub-query and passage by independently retrieving relevant documents. This approach ensures comprehensive consideration of all perspectives in the final retrieval. The MMLF pipeline is illustrated in Figure 1.

2.1 Multi-query Generation

In the first stage, we generate multiple sub-queries from the original query q to capture various aspects of user intent. Using a large language model (LLM) with the `MQR` prompt (detailed in Appendix B.3), we provide sub-queries $q_1, q_2, \ldots, q_n$. This method expands the interpretations of the user's intent, broadening the retrieval scope and increasing the likelihood of discovering relevant information.

2.2 Query-to-passage Expansion

In the second stage, each generated sub-query is expanded into a detailed passage, also referred to as a pseudo-document, enriching the context available for retrieval. We employ an LLM with the `CQE` prompt (detailed in Appendix B.5) to transform each sub-query q_i into corresponding passages p_i. This expansion builds on the first stage by providing a more thorough comprehension of various interpretations identified during the multi-query generation phase.

This two-step process—generating multiple sub-queries and then expanding each into a passage, referred to hereafter as MQ2MP—uniquely separates query decomposition and passage generation. This enables finer-grained control over the retrieval process, achieving results beyond the additive contributions of prior methods (e.g., Query2Doc and LC-MQR).

2.3 Ranked List Fusion

After generating the passages, we treat each one as a separate query for independent document retrieval. These results are then combined with the list from the original query, creating $n + 1$ ranked lists. RRF is then used to fuse them into a final ranked list. Unlike concatenation, which can dilute focus, RRF aggregates results based on their individual rankings. This method prioritizes consistently relevant documents, thereby enhancing overall retrieval effectiveness and mitigating the influence of less relevant content (Cormack et al., 2009). For the complete formula and details on the RRF algorithm, please refer to Appendix A.2.

3 Experiments

This section accesses the performance of our MMLF pipeline, explicitly examining its capabilities in query reformulation and fusion methods. To

	DBPEDIA		FIQA-2018		NFCORPUS		TREC-COVID		TOUCHE-2020	
	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10
Raw Query	73.76	36.06	86.74	35.50	60.72	31.81	40.49	52.61	70.16	13.23
Query2Doc	73.36	39.93	90.05	36.20	65.42	32.47	45.89	73.92	79.47	28.24
CoT	71.78	37.57	88.08	35.25	64.65	30.53	43.19	74.17	78.71	28.19
LC-MQR w/ RRF	74.52	34.16	89.49	34.34	65.11	31.03	42.17	61.24	73.30	18.91
MILL (w/o PRF & MV)	73.48	40.21	88.56	35.05	64.86	31.57	45.16	75.86	80.84	27.73
MMLF	**79.17**	**42.96**	**91.02**	**37.86**	**67.03**	**34.09**	**48.82**	**77.27**	**81.44**	**28.60**

Table 1: Main results

ensure a fair comparison, we omit advanced techniques such as pseudo-relevance feedback, mutual verification (Jia et al., 2024), and few-shot prompting during the evaluation.

3.1 Datasets and Metrics

We evaluate our approach using five low-resource datasets from the BEIR benchmark[2] (Thakur et al., 2021): DB-Pedia, FIQA-2018, NF-Corpus, TREC-COVID, and Touche-2020. These publicly available datasets cover various domains and query complexities, making them well-suited for evaluating retrieval systems, especially in zero-shot scenarios. Additionally, we test larger BEIR datasets, including MS MARCO, NQ, FEVER, and HOTPOTQA, to evaluate scalability (see Appendix E). Our primary evaluation metric is Recall@1k, essential for dense retrieval tasks to identify a wide array of relevant documents for subsequent reranking. We also report nDCG@10, a top-heavy metric that prioritizes higher-ranked relevant documents.

3.2 Experimental Setup

We conduct our experiments using the Llama-3-70B-Instruct model[3]. The model parameters include a temperature setting of 1 and a top_p of 1. For encoding, we employ the e5-small-v2 model[4] (Wang et al., 2022), producing a 384-dimensional embedding for each query and document. We compute the cosine similarity between the embeddings of the queries and the documents for document retrieval, ranking them based on their relevance. While our method supports a flexible number of sub-queries, we fix it at 3 for consistency. Additional experiments using a smaller LLM, Llama-3-8B-Instruct, and an alternative encoder, Contriever[5](Izacard et al., 2021), are detailed in Appendix F and Appendix G, respectively.

[2]Licensed under Apache-2.0.

[3]Available at https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct and licensed under LLaMA3.

[4]Licensed under MIT.

[5]Licensed under CC BY-NC 4.0.

3.3 Baselines and Competitiors

We compare our approach with five baseline methods: **1) Raw Query**, which uses the original query without reformulation; **2) Query2Doc** (Wang et al., 2023), which expands the original query into a passage for retrieval alongside the query; **3) Chain-of-Thought (CoT)** (Jagerman et al., 2023), which transforms the query into an answer and a rationale for retrieval alongside the query; **4) LangChain MultiQueryRetriever (LC-MQR),**[6] generating multiple sub-queries and using reciprocal rank fusion (RRF) for reranking; **5) MILL** (Jia et al., 2024), which creates sub-queries and passages but skips advanced techniques like pseudo-relevance feedback and mutual verification for direct comparison. Detailed descriptions of each pipeline are provided in Appendix A.1.

3.4 Main Results

In this section, we compare our MMLF's performance to the five baselines. As shown in Table 1, MMLF consistently outperforms all baselines in Recall@1k and nDCG@10 across five datasets. Specifically, our approach achieves an average improvement of 4% in Recall@1k over the closest competitor, MILL, demonstrating a substantial gain, particularly given the high performance of existing methods.

3.5 Ablation Study

For each of the ablation experiments, we present results for both Recall@1k and nDCG@10. Due to the page constraints, the nDCG@10 results are detailed in Appendix D. Additional prompt variation analysis is provided in Appendix C.

3.5.1 Fusion Method Comparison

This experiment evaluates the effectiveness of **RRF**, a rank-based late-fusion technique (Cormack

[6]LC-MQR originally combines retrieval results with a unique union, but this produces an unordered set. We replace it with RRF for comparability.

	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
concat w/ q	73.67	87.63	64.72	44.12	78.68	69.76
CombSUM w/ q included	77.53	90.07	66.76	48.71	80.97	72.81
RRF w/ q included	**79.17**	**91.02**	**67.03**	**48.82**	**81.44**	73.50

Table 2: Fusion methods comparison (Recall@1k)

	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
RRF w/o q	74.54	89.47	66.77	47.34	79.43	71.51
RRF w/ q concatenated	75.44	90.08	66.93	47.12	80.01	71.92
RRF w/ q included	79.17	**91.02**	**67.03**	**48.82**	**81.44**	73.50
RRF w/ q included and concatenated	**79.32**	90.84	66.98	48.41	80.67	73.24

Table 3: Impact of including the original query (Recall@1k)

	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
MQ	74.52	89.49	65.11	42.17	73.30	68.92
MP	76.26	89.54	65.38	45.78	79.40	71.27
MQ2MP	**79.17**	**91.02**	**67.03**	**48.82**	**81.44**	73.50

Table 4: Impact of generating passages in two stages (Recall@1k)

et al., 2009), compared to **CombSUM**, a score-based late-fusion method (Fox and Shaw, 1994), and **concatenation**, a basic early-fusion approach. Detailed formulations for each fusion method are provided in Appendix A.2.

As illustrated in Table 2, **RRF** consistently outperforms both **Concatenation** and **CombSUM** in terms of Recall@1k across all datasets. This indicates that rank-based late-fusion strategies like **RRF** are more effective in aggregating diverse retrieval outputs. However, for nDCG@10, **CombSUM** achieves the best results, showcasing its ability to rank the most relevant documents at the top.

3.5.2 Role of the Original Query

This experiment examines the impact of including the original query in the RRF process of the MMLF pipeline. We aim to see if this improves retrieval performance or if other setups are more effective. Detailed formulations for each configuration are available in Appendix A.3. The configurations assessed are as follows:

- **RRF w/o** q: Utilize only the passages for retrieval, excluding the original query.
- **RRF w/** q **concatenated** : Concatenate the original query with each passage prior to retrieval.
- **RRF w/** q **include**: Use the original query and the passages separately for retrieval.
- **RRF w/** q **included and concatenated**: Include the original query both separately and concatenated with each passage before retrieval.

For all four configurations described above, RRF is applied to the resulting retrieval outcomes.

The results in Table 3 show that including the original query alongside passages consistently yields better performance in Recall@1k across most datasets, as seen in the last three configurations compared to the first. Among the last three, the "RRF w/ q included" configuration attains the highest average Recall@1k of 73.50, surpassing the configurations where the query is excluded or concatenated with passages.

3.5.3 Query Reformulation Pipeline

This experiment evaluates the effectiveness of our proposed two-stage query-reformulation pipeline, **MQ2MP**, against other query-passage generation strategies. Detailed descriptions of each pipeline are provided in Appendix A.4. The configurations tested include:

- **MQ:** Uses sub-queries directly for retrieval without transforming them into passages.
- **MP:** Generate passages directly with the original query q using the prompt `MCQE` (see Appendix B.6).
- **MQ2MP:** Generate sub-queries using the prompt `MQR` (see Appendix B.3), then individually expand them into passages using the propmt `CQE` (see Appendix B.5).

Results in Table 4 demonstrate that **MQ2MP** consistently outperforms both **MP** and **MQ** in terms of Recall@1k, achieving the highest average score of 73.50, particularly excelling in the DBPEDIA and TREC-COVID datasets. This highlights a non-trivial result where combining query decomposition and context enrichment preserves the improvements of both approaches on average without diminishing their contributions. While the **MP** approach, which generates passages in a single step, performs moderately well, it still lags behind the more nuanced two-stage **MQ2MP** method. Additionally, while the **MP** method scores reason-

ably on nDCG@10, as detailed in Appendix D (Table 14), it does not surpass the two-stage approach. The two-stage **MQ2MP** method generates diverse contextual passages, akin to the one-stage **MP** approach. However, as demonstrated by the Recall@1k and nDCG@10 results, **MQ2MP** consistently outperforms **MP**, suggesting that the additional step of expanding sub-queries into passages enhances effective retrieval.

4 Conclusion

We introduced MMLF, an effective and robust information retrieval (IR) pipeline that leverages large language models (LLMs) to generate diverse sub-queries and expand them into contextual pseudo-documents. Our experiments demonstrate marked improvements in retrieval performance, particularly in Recall@1k, across multiple datasets without requiring model fine-tuning. MMLF presents a scalable and adaptable solution for improving search performance across various domains, uniquely combining query decomposition and passage generation to advance query reformulation for dense retrieval.

5 Limitations

Despite its effectiveness, MMLF presents several computational challenges. On the generation side, the method requires running inference with LLMs, which can be considerably slow due to token-by-token autoregressive decoding. The need to generate multiple sub-queries and expand each into detailed passages substantially increases the computational load. Although parallel processing of passages can mitigate some of this overhead, it still represents a bottleneck, especially in real-time or large-scale systems.

On the retrieval side, MMLF independently processes and retrieves documents for each expanded passage, increasing the overall retrieval workload. Applying reciprocal rank fusion (RRF) to combine these results adds complexity, particularly when scaling to larger corpora or high query volumes.

Furthermore, although we fixed the number of sub-queries generated to three in our experiments, we did not perform an ablation study to assess how varying the number of sub-queries affects retrieval performance and computational costs. This is an area for future investigation that could explore the trade-offs between query diversity, retrieval effectiveness, and computational demands.

These computational challenges may limit MMLF's applicability in environments constrained by resources or time. Future research could focus on optimizing the LLM inference stage and improving the efficiency of the retrieval and fusion techniques to reduce computational overhead without compromising retrieval effectiveness.

References

J. Bhogal, A. Macfarlane, and P. Smith. 2007. A review of ontology based query expansion. *Information Processing & Management*, 43(4):866–886.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759.

Edward Fox and Joseph Shaw. 1994. Combination of multiple searches. *NIST Special Publication*, pages 243–243.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin,

and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

Pengyue Jia, Yiding Liu, Xiangyu Zhao, Xiaopeng Li, Changying Hao, Shuaiqiang Wang, and Dawei Yin. 2024. MILL: Mutual verification with large language models for zero-shot query expansion. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2498–2518.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Yonggang Qiu and Hans-Peter Frei. 1993. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169.

Zackary Rackauckas. 2024. Rag-fusion: A new take on retrieval augmented generation. *International Journal on Natural Language Computing*, 13(1):37–47.

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 *Proceedings of the 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

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*, pages 9414–9423.

Appendix

A Pipeline Formulations

This appendix provides the formulations of the retrieval pipelines used in our experiments, with a consistent generation of 3 sub-queries or passages across all configurations.

A.1 Baseline Methods

This section details the formulations for the baseline retrieval methods listed in Section 3.3 along with our MMLF. In the description below, $\mathcal{R}(\cdot)$ represents the retriever.

1) **Raw Query:** The original query q is used directly for retrieval:

$$\mathcal{R}(q).$$

2) **Query2Doc (Q2D):** The original query q is expanded into a passage p^{Q2D} using the `Q2D` prompt (see Appendix B.1) and concatenated with q, separated by a [SEP] token, for retrieval:

$$\mathcal{R}\left(\text{concat}\left(q,\ [\text{SEP}]\ , p^{\text{Q2D}}\right)\right).$$

3) **Chain-of-Thought (CoT):** The original query q is expanded into a rationale and answer pair p^{CoT} using the `CoT` prompt (see Appendix B.2) and concatenated with q, separated by a [SEP] token, before retrieval:

$$\mathcal{R}\left(\text{concat}\left(q,\ [\text{SEP}]\ , p^{\text{CoT}}\right)\right).$$

4) **LangChian MultiQueryRetriever (LC-MQR):** Three sub-queries $q_1^{\text{MQR}}, q_2^{\text{MQR}}, q_3^{\text{MQR}}$ are generated from the original query using the `MQR` prompt (see Appendix B.3), and RRF is applied to their retrieval results along with the original query:

$$\text{RRF}\left(\mathcal{R}(q), \mathcal{R}\left(q_1^{\text{MQR}}\right), \mathcal{R}\left(q_2^{\text{MQR}}\right), \mathcal{R}\left(q_3^{\text{MQR}}\right)\right).$$

5) **MILL:** The original query q is reformulated into three passages $p_1^{\text{QQD}}, p_2^{\text{QQD}}, p_3^{\text{QQD}}$ using the `QQD` prompt (see Appendix B.4), and all passages are concatenated with q, separated by [SEP] tokens, before retrieval:[7]

$$\mathcal{R}\Big(\text{concat}\Big(q, [\text{SEP}]\ , p_1^{\text{QQD}}, [\text{SEP}]\ , p_2^{\text{QQD}}, [\text{SEP}]\ , p_3^{\text{QQD}}\Big)\Big).$$

[7] Note that we adopt the approach used in MILL (Jia et al., 2024), where the generated sub-quries are not used in the retrieval process. Note that in our experiments, we skip advanced techniques like pseudo-relevance feedback and mutual verification for direct comparison.

6) **MMLF (ours):** Sub-queries $q_1^{\text{MQR}}, q_2^{\text{MQR}}, q_3^{\text{MQR}}$ are first generated using the `MQR` prompt (see Appendix B.3), then each is expanded into passages $p_1^{\text{MQR-CQE}}, p_2^{\text{MQR-CQE}}, p_3^{\text{MQR-CQE}}$ using the `CQE` prompt (see Appendix B.5). RRF is applied to the retrieval results of the original query and the expanded passages:

$$\text{RRF}\Big(\mathcal{R}(q), \mathcal{R}\Big(p_1^{\text{MQR-CQE}}\Big), \mathcal{R}\Big(p_2^{\text{MQR-CQE}}\Big), \mathcal{R}\Big(p_3^{\text{MQR-CQE}}\Big)\Big).$$

A.2 Fusion Method Comparison

This section details the formulations of the fusion methods listed in Section 3.5.1. The formulations provided below use the original query q and passages p_1, p_2, p_3 as examples, but these fusion methods are also applicable in other contexts, such as with sub-queries or various combinations of queries and passages.

- **Concatenation:** In this early-fusion approach, the original query and passages are concatenated into a single sequence, with `[SEP]` tokens separating each part. The retrieval function, $\mathcal{R}$, is then applied to this concatenated sequence. The formulation is as follows:

$$\mathcal{R}\big(\text{concat}(q, [\text{SEP}]\ , p_1, [\text{SEP}]\ , p_2, [\text{SEP}]\ , p_3)\big).$$

- **CombSUM:** This score-based late-fusion method aggregates the similarity scores from the retrieval results of both the original query and the individual passages. The retrieval function $\mathcal{R}(\cdot)$ returns a ranked list of documents, each with a corresponding similarity score. The final score for each document d is given by:

$$\text{score}_{\text{CombSUM}}(d) = \sum_{i=0}^{3} \text{score}_i(d),$$

where $\text{score}_i(d)$ is the similarity score of document d from the retrieval result $\mathcal{R}(p_i)$ for $i = 1, 2, 3$ and for $i = 0$, the score comes from the retreival results $\mathcal{R}(q)$ of the original query q. The formulation is as follows:

$$\text{CombSUM}\left(\mathcal{R}(q), \mathcal{R}(p_1), \mathcal{R}(p_2), \mathcal{R}(p_3)\right),$$

aggregating the retrieval results of the original query and the passages by summing their scores.

- **Reciprocal Rank Fusion (RRF):** This rank-based late-fusion method aggregates retrieval results based on the ranks of documents instead of their similarity scores. The retrieval function $\mathcal{R}(p_i)$ provides a ranked list of documents for each query or passage. The combined score for each document d is calculated as follows:

$$\text{score}_{\text{RRF}}(d) = \sum_{i=0}^{3} \frac{1}{k + \text{rank}_i(d)},$$

where $\text{rank}_i(d)$ is the position of document d in the resulting ranked list $\mathcal{R}(p_i)$, and $k = 60$ is a constant typically used in RRF implementations. The formulation is as follows:

$$\text{RRF}\left(\mathcal{R}(q), \mathcal{R}(p_1), \mathcal{R}(p_2), \mathcal{R}(p_3)\right),$$

which aggregates the ranks of the original query and passages to produce a final ranked list.

A.3 Role of the Original Query

This section details the formulations of the configurations in Section 3.5.2, which evaluates the impact of including or excluding the original query in the retrieval process.

- **RRF w/o** q: In this configuration, only the passages are used for retrieval, excluding the original query:

$$\text{RRF}\left(\mathcal{R}(p_1), \mathcal{R}(p_2), \mathcal{R}(p_3)\right).$$

- **RRF w/** q **concatenated**: The original query is concatenated with each passage before retrieval:

$$\begin{aligned}\text{RRF}\big(&\mathcal{R}(\text{concat}(q,\ [\text{SEP}]\ , p_1)),\\ &\mathcal{R}(\text{concat}(q,\ [\text{SEP}]\ , p_2)),\\ &\mathcal{R}(\text{concat}(q,\ [\text{SEP}]\ , p_3))\big).\end{aligned}$$

- **RRF w/** q **include**: The original query is included alongside the passages, with RRF applied to the retrieval results:

$$\text{RRF}\left(\mathcal{R}(q), \mathcal{R}(p_1), \mathcal{R}(p_2), \mathcal{R}(p_3)\right).$$

- **RRF w/** q **included and concatenated**: The original query is both included separately and concatenated with each passage:

$$\begin{aligned}\text{RRF}\big(&\mathcal{R}(q),\\ &\mathcal{R}(\text{concat}(q,\ [\text{SEP}]\ , p_1)),\\ &\mathcal{R}(\text{concat}(q,\ [\text{SEP}]\ , p_2)),\\ &\mathcal{R}(\text{concat}(q,\ [\text{SEP}]\ , p_3))\big).\end{aligned}$$

A.4 Query Reformulation Pipeline

This section details the configurations in Section 3.5.3, where we compare the performance of different approaches for generating and expanding sub-queries. The approaches include:

- **MQ:** In this approach, sub-queries are generated from the original query q using the `MQR` prompt (see Appendix B.3). The resulting sub-queries $q_1^{\text{MQR}}, q_2^{\text{MQR}}, q_3^{\text{MQR}}$ are used for retrieval without any passage expansion. The retrieval process is conducted as follows:

$$\begin{aligned}\text{RRF}\Big(&\mathcal{R}(q), \mathcal{R}\left(q_1^{\text{MQR}}\right),\\ &\mathcal{R}\left(q_2^{\text{MQR}}\right),\\ &\mathcal{R}\left(q_3^{\text{MQR}}\right)\Big).\end{aligned}$$

- **MP:** This approach directly generates passages from the original query q using the `MCQE` prompt (see Appendix B.6), bypassesing sub-query generation. It produces three passages $p_1^{\text{MCQE}}, p_2^{\text{MCQE}}, p_3^{\text{MCQE}}$. The retrieval process is conducted as follows:

$$\begin{aligned}\text{RRF}\Big(&\mathcal{R}(q), \mathcal{R}\left(p_1^{\text{MCQE}}\right),\\ &\mathcal{R}\left(p_2^{\text{MCQE}}\right),\\ &\mathcal{R}\left(p_3^{\text{MCQE}}\right)\Big).\end{aligned}$$

- **MQ2MP:** This two-stage pipeline first generates sub-queries $q_1^{\text{MQR}}, q_2^{\text{MQR}}, q_3^{\text{MQR}}$ using the `MQR` prompt (see Appendix B.3). Each sub-query is then expanded into a corresponding passage $p_1^{\text{MQR-CQE}}, p_2^{\text{MQR-CQE}}, p_3^{\text{MQR-CQE}}$ using the `CQE` prompt (see Appendix B.5). The retrieval is then performed as follows:

$$\begin{aligned}\text{RRF}\Big(&\mathcal{R}(q), \mathcal{R}\left(p_1^{\text{MQR-CQE}}\right),\\ &\mathcal{R}\left(p_2^{\text{MQR-CQE}}\right),\\ &\mathcal{R}\left(p_3^{\text{MQR-CQE}}\right)\Big).\end{aligned}$$

B Prompt Formulations

This section describes the prompts used in the query reformulation process. We incorporate specific instructions in our prompt to guide the LLMs in formatting their outputs, which aids in easier parsing and minimizes the generation of irrelevant text. In each prompt, we denote the query to be reformulated as {query}.

B.1 Query2doc Prompt (Q2D)

The Q2D prompt, originating from (Wang et al., 2023), is designed to expand a user query into a passage that directly addresses the query. This method effectively bridges the gap between short queries and the detailed information required for effective document retrieval, as illustrated in Table 5.

Q2D prompt
Please write a passage to answer the query. Query: {query} Format your response in plain text as: Passage:

Table 5: The Q2D prompt

B.2 Chain-of-Thought Prompt (CoT)

The original CoT prompt instructs the LLM to provide a rationale before answering the query. To prevent introductory text from being parsed, we modify the original prompt to explicitly structure the response with labelled sections for the rationale and answer, as shown in Table 6. These sections are then concatenated with "\n\" to ensure clarity during retrieval.

CoT prompt
Answer the following query: Query: {query} Provide the rationale before answering, and format your response in plain text as: Rationale: Answer:

Table 6: The CoT prompt

B.3 LangChain MultiQueryRetriever Prompt (MQR)

The MQR prompt generates sub-queries by offering explicit task instructions and context to provide multiple perspectives on a user question, enhancing retrieval effectivness.[8] We adapt this prompt to generate three sub-queries, using a more structured format than the original one used in LangChain MultiQueryRetriever, as shown in Table 7.

MQR prompt
You are an AI language model assistant. Your task is to generate exactly three different versions of the given user question to retrieve relevant documents from a vector database. By generating multiple perspectives on the user question, your goal is to help the user overcome some of the limitations of the distance-based similarity search. Original question: {query} Format your response in plain text as: Sub-query 1: Sub-query 2: Sub-query 3:

Table 7: The MQR prompt

B.4 Query-Query-Document Prompt (QQD)

The QQD prompt implements the query-query-document approach from MILL (Jia et al., 2024), with adaptations to generate both sub-queries and corresponding passages, as detailed in Table 8,

QQD prompt
Generate exactly three different versions of the given user question to retrieve relevant documents from a vector database. For each sub-query, also write a passage that answers it. The goal is to provide varied perspectives to enhance the effectiveness of similarity search. Original question: {query} Format your response in plain text as: Sub-query 1: Passage 1: Sub-query 2: Passage 2: Sub-query 3: Passage 3:

Table 8: The QQD prompt

[8] https://api.python.langchain.com/en/latest/retrievers/langchain.retrievers.multi_query.MultiQueryRetriever.html

B.5 Combined Query Expansion Prompt (CQE)

We design the CQE prompt, as shown in Table 9, specifically for our pipeline. It generates a passage that simultaneously addresses both the original query and a sub-query, maintaining the diversity introduced by the sub-queries while ensuring alignment with the original query.

CQE prompt
Please write a passage to answer the following user questions simultaneously. Question 1: {original_query} Question 2: {sub_query} Format your response in plain text as: Passage:

Table 9: The CQE prompt

B.6 Multiple Combined Query Expansion prompt (MCQE)

The MCQE prompt, as detailed in Table 10, integrates multi-query generation and query-to-passage expansion into a single step. It generates three sub-queries with the original query and crafts a passage for each sub-query, addressing both the original query and its respective sub-query. This approach is designed to capture diverse perspectives while ensuring relevance to the original query.

MCQE prompt
You are an AI language model assistant. Your task is to generate exactly three different versions of the given user question (sub-queries) and then write a passage for each sub-query to retrieve relevant documents from a vector database. Each passage should address both the original query and its corresponding sub-query. By generating multiple passages from different perspectives, your goal is to help the user overcome some of the limitations of distance-based similarity search. Original question: {query} Format your response in plain text as: Sub-query 1: Passage 1: Sub-query 2: Passage 2: Sub-query 3: Passage 3:

Table 10: The MCQE prompt

C Impact of Prompts

This experiment evaluates the impact of various prompt variations on the performance of our two-stage query reformulation pipeline. We evaluate the original two-stage setup against alternatives using different prompts for generating sub-queries and expanding passages. The variations tested include:

- **MQR with CQE:** Sub-queries are first generated with multiQ and then expanded into passages using CQE.
- **MQR with Q2D:** Instead of CQE, sub-queries are expanded using Q2D.
- **MQR with CoT:** Sub-queries are expanded using the CoT prompt instead of CQE.

As illustrated in Table 11, the one with CQE mostly outperforms the others, indicating that addressing both the original query and sub-query simultaneously can enhance retrieval outcomes. This finding highlights the importance of prompt design in optimizing retrieval performance. Note that the prompt formulations can be found in Appendix B.

D Results in nDCG@10

This section presents detailed nDCG@10 results for the various experimental settings discussed in the paper. These experiments assess different fusion methods, the significance of including the original query, the effectiveness of the query reformulation pipeline, and the influence of different prompt variations, and their results in nDCG@10 are tabulated in Tables 12, 13, 14, and 15, respectively.

E Results on Larger BEIR Datasets

Expanding the evaluation of MMLF, we test its performance on larger datasets from the BEIR benchmark (Thakur et al., 2021), including MS MARCO, NQ, FEVER, and HOTPOTQA. MS MARCO and NQ serve as general-purpose datasets with broad retrieval contexts, while all four datasets feature substantial query-document scales, making them valuable additions to the evaluation. Note that MS MARCO and NQ have been used to fine-tune the e5-small-v2 (Wang et al., 2022) model, which serves as our main retriever, potentially influencing

Method	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
MQR & Q2D	78.25	90.39	**67.60**	**48.94**	80.39	73.11
MQR & CoT	72.93	89.09	65.78	47.01	80.45	71.05
MQR & CQE	**79.17**	**91.02**	67.03	48.82	**81.44**	73.50

Table 11: Impact of different prompt variations (Recall@1k)

Method	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
concat w/ q	40.48	34.32	34.14	74.07	28.55	42.31
CombSUM w/ q included	**43.65**	**38.61**	**34.97**	**79.25**	**29.77**	45.25
RRF w/ q included	42.96	37.86	34.09	77.27	28.60	44.16

Table 12: Fusion methods comparison (nDCG@10)

performance on these datasets. The results, shown in Table 16, indicate that MMLF performs consistently well across these settings, demonstrating its scalability and effectiveness in handling large-scale retrieval tasks.

F Results with Llama-3-8B-Instruct

To examine the impact of LLM size on the performance of query reformulation methods, we conduct experiments using Llama-3-8B-Instruct[9], a scaled-down variant of Llama-3-70B-Instruct. This evaluation helps assess whether smaller LLMs can serve as viable alternatives in scenarios with limited computational resources. As shown in Table 17, query reformulation methods, including MMLF, continue to outperform raw queries in most cases, although performance decreases compared to larger models such as Llama-3-70B-Instruct. Notably, MMLF is the only method to improve Recall@1k over the raw query baseline on DBPEDIA and consistently achieves the highest or near-highest scores across all datasets. Furthermore, MMLF narrows the performance gap between smaller and larger LLMs, outperforming Q2D with Llama-3-70B-Instruct on average across both Recall@1k and nDCG@10. These findings suggest that smaller LLMs, despite their limitations, remain effective for retrieval tasks, particularly when computational efficiency is a priority.

G Results with Contriever

Building on the positive outcomes obtained with the e5-small-v2 (Wang et al., 2022) encoder, we further investigate the performance enhancement of using the Contriever (Izacard et al., 2021) as the underlying model. As shown in Table 18, the MMLF method demonstrates notable performance gains across almost all datasets, mirroring the advancements seen with e5-small-v2. These consistent gains validate the effectiveness of MMLF in enhancing retrieval through query augmentation, significantly boosting metrics such as nDCG@10 and Recall@1k. This improvement suggests that MMLF effectively leverages in-domain knowledge to refine queries, ensuring better alignment between user queries and relevant documents across different retrieval architectures.

[9]Available at https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct and licensed under LLaMA3.

Method	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
RRF w/o q	41.15	34.09	33.85	73.32	27.72	42.03
RRF w/ q included	42.96	37.86	**34.09**	77.27	28.60	44.16
RRF w/ q concatenated	41.81	37.83	33.63	76.04	**31.12**	44.09
RRF w/ q included and concatenated	**43.02**	**39.64**	33.91	**78.01**	29.31	44.78

Table 13: Impact of including the original query (nDCG@10)

Method	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
MQ	34.16	34.34	31.03	61.24	18.91	35.94
MP	39.15	34.67	32.96	76.91	24.62	41.66
MQ2MP	**42.96**	**37.86**	**34.09**	**77.27**	**28.60**	44.16

Table 14: Impact of generating passages in two stages (nDCG@10)

Method	DBPEDIA	FIQA-2018	NFCORPUS	TREC-COVID	TOUCHE-2020	Avg.
MQR & Q2D	40.46	34.78	31.91	76.63	27.22	42.20
MQR & CoT	32.50	32.71	31.09	75.95	25.85	39.62
MQR & CQE	**42.96**	**37.86**	**34.09**	**77.27**	**28.60**	44.16

Table 15: Impact of different prompt variations (nDCG@10)

Method	MS MARCO		NQ		FEVER		HOTPOTQA	
	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10
Raw Query	77.48	68.72	98.14	52.15	97.30	69.98	90.82	64.99
Query2Doc	83.02	73.06	98.55	59.99	98.00	76.60	93.54	70.81
CoT	81.48	71.05	97.88	56.32	97.12	69.41	93.33	72.07
LC-MQR w/ RRF	80.02	69.15	98.73	53.15	97.75	68.16	91.65	64.75
MILL (w/o PRF & MV)	79.95	65.12	98.01	55.31	97.24	74.74	93.47	71.52
MMLF	**86.17**	**73.72**	**99.47**	**64.01**	**98.45**	**78.62**	**95.11**	**73.21**

Table 16: Results on larger BEIR datasets

Method	DBPEDIA		FIQA-2018		NFCORPUS		TREC-COVID		TOUCHE-2020	
	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10
Raw Query	73.76	36.06	86.74	35.50	60.72	31.81	40.49	52.61	70.16	13.23
Query2Doc	72.16	37.99	88.22	**36.91**	64.26	31.60	44.31	75.75	77.48	28.26
CoT	70.12	35.22	87.88	34.35	63.28	30.29	43.46	72.33	**80.95**	23.79
LC-MQR w/ RRF	72.49	30.48	87.84	32.66	63.19	28.80	41.13	65.29	76.01	19.49
MILL (w/o PRF & MV)	69.73	36.16	85.78	32.17	64.67	29.03	41.33	71.28	75.36	26.07
MMLF	**78.36**	**42.13**	**89.93**	36.66	**65.56**	**32.81**	**48.43**	**78.00**	79.58	**30.74**

Table 17: Results with llama-3-8b-instruct

Method	DBPEDIA		FIQA-2018		NFCORPUS		TREC-COVID		TOUCHE-2020	
	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10	Recall@1k	nDCG@10
Raw Query	71.42	29.16	82.15	24.50	62.34	31.73	16.75	27.45	71.44	16.68
Query2Doc	69.48	32.30	85.31	26.03	64.60	28.63	32.86	56.22	74.53	20.87
CoT	66.82	29.67	83.83	25.05	63.68	29.15	34.12	54.01	73.46	23.11
LC-MQR w/ RRF	73.10	27.86	84.20	21.98	65.83	29.58	19.78	29.12	70.81	12.76
MILL (w/o PRF & MV)	69.19	32.24	85.39	25.40	64.78	27.61	**34.15**	**58.15**	74.65	22.94
MMLF	**74.54**	**35.69**	**88.09**	**29.47**	**66.16**	**31.30**	31.89	56.17	**76.87**	**24.25**

Table 18: Results with Contriever