LLM-Personalized Retrieval: Query Rewriting and Negatives for Session Search

Szu-Ju Chen
B10705005 b10705005@ntu.edu.tw
Model Training, Negative Data Process

Yi-Shin Chiu
B10705009 b10705009@ntu.edu.tw
Data Construction, Query Rewrite

1 Introduction

Traditional search and retrieval systems often rely on static user profiles or past interactions, which may fail to capture rapidly changing user intents—particularly in multi-turn or session-based search scenarios. With the recent advancements in Large Language Models (LLMs), there is an opportunity to better understand user preferences and session context dynamically, without costly manual intervention. In this project, we leverage the capabilities of LLMs to perform query rewriting using user behavior information, with the goal of generating more personalized queries that reflect the user's intent more accurately. Additionally, we explore the impact of incorporating negative document examples during the training of the retrieval model, aiming to improve its overall performance.
Our objectives are:
(1) Investigate the effectiveness of LLM-generated query rewrites in enhancing personalized search
(2) Examine whether hard negatives can further boost the relevance and accuracy of the retrieved results.

2 Related Work

With the growing advancements in Large Language Models (LLMs), many recent studies have focused on leveraging LLMs to generate training data.

Liu et al.[4] introduces natural language rewrite rules to enhance complex query reformulation, aiming to make rewritten queries more interpretable and effective. Sun et al.[8] applies a rule-based and step-by-step prompt strategy to guide LLMs in generating high-quality rewritten queries, achieving significant improvements. Sinha et al.[7] explores query and negative sample generation for retrieval model training and demonstrates competitive performance. These approaches highlight the utility of LLM-generated data in improving retrieval models while reducing the need for manual annotations.

In addition to query rewriting, several works have explored the effectiveness of incorporating negative samples during training. Zhan et al.[12] provide a comprehensive analysis of negative sampling strategies and their impact on retrieval performance. Nguyen et al.[5] propose selecting negatives based on their similarity to positive documents rather than queries, using a Vietnamese dataset. Yang et al.[11] suggest sampling more informative negatives from a constrained region in the embedding space to boost model.

3 Methodology

3.1 Data Construction

In this project, we use the TREC 2014 Session Track[2] dataset for both training and evaluation. This dataset consists of multiple search sessions from different users. Each session contains a topic description and a sequence of user interactions. Each interaction includes a query, the corresponding retrieved search results, and optionally, a set of user-clicked documents. To construct training samples, we extract (query, clicked document) pairs by mapping each query to the first user-clicked document within the same interaction. There are total 1041 pairs, 936 samples for training and 105 samples for testing. For the query rewriting process, we further extract and provide the session topic, previously issued queries, and previously clicked documents as additional context for the LLMs. This contextual information helps the model better understand user intent and generate more personalized rewritten queries.

3.2 Query Rewrite

We applied a query rewriting strategy using two open-source LLMs: LLaMA[9] and Mistral[3]. In our initial approach, we directly provided the contextual information—including the session topic, previous queries, and clicked documents—and prompted the LLM to rewrite the current query. The system prompt and an example of the output are illustrated in Fig. A. Upon examining the rewritten queries, we observed that they often contained redundant information or irrelevant details. To address this issue, we refined our strategy in two ways: (1) constraining the length of the generated query and (2) instructing the LLM to generate only relevant keywords. The updated system prompts and example outputs for these two approaches are shown in Fig. A and Fig. A, respectively. As shown in the figures, these refinements led to more concise and focused queries that better aligned with the user's current intent.

3.3 Retrieval Model

After data preprocessing, we implemented our retrieval system using a traditional BiEncoder architecture. We experimented with two pretrained encoders: `sentence-transformers/msmarco-MiniLM-L-6-v3` [6], and `BAAI/bge-base-en-v1.5` [10].

In this setup, both queries and documents are independently encoded into dense vector representations. We then compute the cosine similarity between the query embedding and all document embeddings in the corpus to retrieve the top-ranked documents. For training, we employed two types of loss functions. When only positive query-document pairs were available, we used cross-entropy loss. When both positive and negative samples were present, we applied the InfoNCE loss to encourage the model to distinguish relevant documents from hard negatives more effectively.

3.4 Negative Data

To improve the retrieval performance, we incorporated negative training samples using two mining strategies: static BM25-based negatives and dynamically mined negatives from the training BiEncoder.

For the static approach, we used the BM25 [1] algorithm to retrieve top-ranked documents for each query. The highest-ranked document that is not the ground-truth positive was selected as the hard negative example for that query.

In the dynamic approach, we began training with the BM25-based negatives during the initial warm-up epochs. Afterward, we switched to using the current state of the BiEncoder model to retrieve hard negatives. This dynamic mining strategy ensures that the negative samples better reflect the model's evolving understanding, allowing it to learn to distinguish more challenging and semantically similar documents.

4 Experiments

Based on the preprocessing strategies, LLM-based query rewriting, and training settings described above, we conducted retrieval experiments using both the original queries and three variants of rewritten queries generated by two different LLMs. The retrieval evaluation was performed using the baseline BM25 and two trained BiEncoder models, with and without the use of negative data mining strategies.

4.1 Evaluation Metrics

To evaluate the performance of our retrieval system, we use the following standard information retrieval metrics, computed based on the ranking of the single positive document among the retrieved results:

- **Hit@k**: Measures whether the positive document appears within the top-k retrieved results. It is a binary indicator of success.
- **MRR@k** : Evaluates how early the positive document appears in the ranking by computing the reciprocal of its rank, averaged over all queries.
- **NDCG@k** : Assesses the ranking quality by assigning a relevance score of 1 to the positive document and 0 to all others, then computing the DCG and normalizing it by the ideal DCG.

4.2 Rewrite Prompt Template

We evaluated the effectiveness of different query rewriting strategies using BM25 as a baseline retrieval model. As shown in Table1, appending keywords generated by LLaMA to the user's original query consistently improves retrieval performance across all three metrics.

In contrast, rewriting the entire query with LLMs results in a slight performance drop compared to using the original query. We attribute this to the fact that the generated queries often include redundant words, such as adjectives or conjunctions, which may introduce noise and distract the retrieval model, ultimately affecting the similarity computation between the query and the document corpus.

4.3 LLMs

Building on the effectiveness of the rewritten keyword strategy, we further analyzed the performance of two different Large Language Models: LLaMA and Mistral. Both models were used to generate keyword-based query rewrites, which were then used to train and evaluate the retrieval systems.

As shown in Table2, the keywords generated by LLaMA consistently outperform or match the performance of those generated by Mistral across most scenarios. The only exception occurs at $k = 10$ with BM25, where Mistral slightly outperforms LLaMA. This suggests that, overall, LLaMA is more effective in producing concise and relevant keywords that enhance retrieval performance.

4.4 BiEncoder Models

As shown in Table3 and Table4, we present the retrieval performance across all experimental settings for both the BGE and MiniLM encoders. The "Improvement" column in each table represents the average difference in performance metrics compared to the baseline BM25 model.

We observe that all configurations using the BGE encoder show consistent performance improvements across various query versions. Notably, the rewritten keyword strategy using LLaMA achieves the best results, even outperforming the original query. This suggests that BGE effectively leverages the semantic richness of the query rewrites.

For the MiniLM encoder, the performance is slightly lower than BGE overall but still competitive. In most cases,

Query Version	Hit@1	Hit@5	Hit@10	MRR@1	MRR@5	MRR@10	NDCG@1	NDCG@5	NDCG@10
Original Query	0.1238	**0.3524**	**0.4476**	0.1238	0.2098	0.2224	0.1238	**0.2455**	**0.2761**
Rewrite Long Query (LLaMA)	0.1048	0.2476	0.2952	0.1048	0.1538	0.1594	0.1048	0.1770	0.1916
Rewrite Long Query (Mistral)	0.1048	0.2667	0.3238	0.1048	0.1678	0.1754	0.1048	0.1926	0.2111
Rewrite Short Query (LLaMA)	0.1143	0.2000	0.3143	0.1143	0.1479	0.1642	0.1143	0.1610	0.1990
Rewrite Short Query (Mistral)	0.1143	0.2667	0.3810	0.1143	0.1621	0.1778	0.1143	0.1877	0.2251
Rewrite Keyword (LLaMA)	**0.1524**	0.3143	0.4095	**0.1524**	**0.2184**	**0.2302**	**0.1524**	0.2425	0.2724
Rewrite Keyword (Mistral)	0.1429	0.3048	**0.4476**	0.1429	0.2041	0.2233	0.1429	0.2292	0.2756

Table 1: BM25 Performance on Different Query Versions

Model	Query Version	Hit@1	Hit@5	Hit@10	MRR@1	MRR@5	MRR@10	NDCG@1	NDCG@5	NDCG@10
BM25	Rewrite Keyword (LLaMA)	**0.1524**	**0.3143**	0.4095	**0.1524**	**0.2184**	**0.2302**	**0.1524**	**0.2425**	0.2724
	Rewrite Keyword (Mistral)	0.1429	0.3048	**0.4476**	0.1429	0.2041	0.2233	0.1429	0.2292	**0.2756**
BGE	Rewrite Keyword (LLaMA)	**0.2476**	**0.4476**	**0.5619**	**0.2476**	**0.3221**	**0.3373**	**0.2476**	**0.3535**	**0.3904**
	Rewrite Keyword (Mistral)	0.2095	0.3714	0.4571	0.2095	0.2681	0.2796	0.2095	0.2938	0.3216
MiniLM	Rewrite Keyword (LLaMA)	**0.1714**	0.3619	0.4762	**0.1714**	**0.2479**	**0.2621**	**0.1714**	**0.2767**	**0.3126**
	Rewrite Keyword (Mistral)	0.1429	0.3619	0.4762	0.1429	0.2251	0.2417	0.1429	0.2592	0.2975

Table 2: LLMs Rewrite Performance on Rewrite Keyword

MiniLM performs evenly or better than BM25. These results confirm that both encoders, when used within the BiEncoder architecture, provide effective and robust solutions for document retrieval in a session-based setting.

4.5 Negative Data Strategy

The performance of retrieval models trained with negative documents per query is displayed in Table3 and Table4. From the results, we observe consistent performance improvements across all configurations that incorporate static or dynamic negative data mining.

In particular, the dynamic hard negative strategy yields the most significant gains. The LLaMA keyword rewrite approach shows the greatest improvement under this setting, achieving an average performance boost of 0.13 with the BGE encoder and 0.11 with the MiniLM encoder. This is a significant enhancement, bringing the rewritten queries to a performance level comparable to or even surpassing that of the original queries which is something not achieved when training solely on query-positive pairs.

Moreover, the comparison between static and dynamic negative mining demonstrates the superior effectiveness of dynamically mined negatives, which better align with the model's evolving decision boundary during training.

5 Conclusion

In this project, we explored various query rewriting strategies using two Large Language Models (LLMs), LLaMA and Mistral, and integrated them into a retrieval system based on BiEncoder architectures. We also investigated the impact of incorporating hard negative samples—both static and dynamically mined—into the training process. Based on the evaluation results, we summarize the following key findings:
(1) LLMs demonstrate the ability to generate meaningful queries by leveraging session context and user intent.
(2) The effectiveness of query rewriting heavily depends on well-designed prompts tailored to the retrieval task.
(3) Expanding queries with relevant keywords is more effective than fully rewriting queries, as it reduces noise and preserves the original intent.
(4) Incorporating negative documents during training significantly improves model performance across all metrics.
(5) Dynamically mined hard negatives further enhance performance by helping the model adapt progressively, leading to the best results.

Query Version	Hit@1	Hit@5	Hit@10	MRR@1	MRR@5	MRR@10	NDCG@1	NDCG@5	NDCG@10	Improv.
Original Query	0.2381	0.4571	0.5048	0.2381	0.3192	0.3257	0.2381	0.3536	0.3691	+0.10
Original Query (static HN)	0.2000	0.3905	0.5143	0.2000	0.2741	0.2923	0.2000	0.3034	0.3451	+0.07
Original Query (dynamic HN)	0.2571	0.4381	0.5333	0.2571	0.3184	0.3318	0.2571	0.3480	0.3795	**+0.11**
Rewrite Long Query (LLaMA)	0.1238	0.2571	0.3810	0.1238	0.1635	0.1798	0.1238	0.1862	0.2260	+0.03
Rewrite Long Query (Mistral)	0.1619	0.3048	0.3905	0.1619	0.2106	0.2216	0.1619	0.2339	0.2612	+0.05
Rewrite Short Query (LLaMA)	0.1619	0.3333	0.4381	0.1619	0.2163	0.2299	0.1619	0.2450	0.2784	+0.08
Rewrite Short Query (LLaMA + static HN)	0.1714	0.3333	0.4095	0.1714	0.2251	0.2351	0.1714	0.2517	0.2762	+0.08
Rewrite Short Query (LLaMA + dynamic HN)	0.1810	0.3238	0.4190	0.1810	0.2273	0.2409	0.1810	0.2511	0.2827	**+0.08**
Rewrite Short Query (Mistral)	0.1714	0.2952	0.4095	0.1714	0.2178	0.2335	0.1714	0.2371	0.2745	+0.05
Rewrite Short Query (Mistral + static HN)	0.1524	0.3524	0.4571	0.1524	0.2275	0.2415	0.1524	0.2586	0.2925	+0.06
Rewrite Short Query (Mistral + dynamic HN)	0.1714	0.2952	0.4762	0.1714	0.2140	0.2390	0.1714	0.2339	0.2933	+0.06
Rewrite Keyword (LLaMA)	0.2476	0.4476	0.5619	0.2476	0.3221	0.3373	0.2476	0.3535	0.3904	+0.11
Rewrite Keyword (LLaMA + static HN)	0.2286	0.4476	0.5333	0.2286	0.3089	0.3219	0.2286	0.3434	0.3726	+0.10
Rewrite Keyword (LLaMA + dynamic HN)	0.2857	0.4190	0.5619	0.2857	0.3352	0.3546	0.2857	0.3562	0.4027	**+0.13**
Rewrite Keyword (Mistral)	0.2095	0.3714	0.4571	0.2095	0.2681	0.2796	0.2095	0.2938	0.3216	+0.06
Rewrite Keyword (Mistral + static HN)	0.1524	0.3810	0.4952	0.1524	0.2249	0.2389	0.1524	0.2631	0.2987	+0.03
Rewrite Keyword (Mistral + dynamic HN)	0.2190	0.4286	0.5238	0.2190	0.2946	0.3064	0.2190	0.3279	0.3578	+0.09

Table 3: BGE BiEncoder Performance on Different Settings (Improv.: compares to the BM25 baseline)

Query Version	Hit@1	Hit@5	Hit@10	MRR@1	MRR@5	MRR@10	NDCG@1	NDCG@5	NDCG@10	Improv.
Original Query	0.1714	0.4762	0.6190	0.1714	0.2710	0.2882	0.1714	0.3214	0.3657	+0.08
Original Query (static HN)	0.2286	0.4762	0.5524	0.2286	0.3184	0.3284	0.2286	0.3577	0.3821	+0.11
Original Query (dynamic HN)	0.2762	0.4857	0.6095	0.2762	0.3519	0.3691	0.2762	0.3852	0.4259	**+0.15**
Rewrite Long Query (LLaMA)	0.0952	0.2571	0.3048	0.0952	0.1514	0.1576	0.0952	0.1776	0.1927	0.00
Rewrite Long Query (Mistral)	0.1238	0.2476	0.3048	0.1238	0.1702	0.1776	0.1238	0.1895	0.2078	0.00
Rewrite Short Query (LLaMA)	0.1429	0.2571	0.3905	0.1429	0.1856	0.2025	0.1429	0.2034	0.2457	+0.04
Rewrite Short Query (LLaMA + static HN)	0.1524	0.2476	0.3905	0.1524	0.1884	0.2070	0.1524	0.2032	0.2490	+0.05
Rewrite Short Query (LLaMA + dynamic HN)	0.1810	0.3333	0.4095	0.1810	0.2348	0.2446	0.1810	0.2592	0.2836	**+0.09**
Rewrite Short Query (Mistral)	0.1429	0.2571	0.4095	0.1429	0.1814	0.2023	0.1429	0.2000	0.2498	+0.02
Rewrite Short Query (Mistral + static HN)	0.1429	0.2476	0.3810	0.1429	0.1732	0.1903	0.1429	0.1913	0.2338	+0.01
Rewrite Short Query (Mistral + dynamic HN)	0.1714	0.2857	0.390	0.1714	0.2122	0.2267	0.1714	0.2304	0.2649	+0.04
Rewrite Keyword (LLaMA)	0.1714	0.3619	0.4762	0.1714	0.2479	0.2620	0.1714	0.2767	0.3126	+0.03
Rewrite Keyword (LLaMA + static HN)	0.1714	0.3714	0.4667	0.1714	0.2498	0.2615	0.1714	0.2804	0.3102	+0.03
Rewrite Keyword (LLaMA + dynamic HN)	0.2667	0.4095	0.5333	0.2667	0.3192	0.3345	0.2667	0.3416	0.3804	**+0.11**
Rewrite Keyword (Mistral)	0.1429	0.3619	0.4762	0.1429	0.2251	0.2417	0.1429	0.2592	0.2975	+0.02
Rewrite Keyword (Mistral + static HN)	0.2000	0.3143	0.4476	0.2000	0.2311	0.2484	0.2000	0.2513	0.2939	+0.03
Rewrite Keyword (Mistral + dynamic HN)	0.2095	0.4000	0.5810	0.2095	0.2711	0.2968	0.2095	0.3029	0.3629	+0.08

Table 4: MiniLM BiEncoder Performance on Different Settings (Improv.: compares to the BM25 baseline)

References

[1] Brown, D. Rank-BM25: A Collection of BM25 Algorithms in Python, 2020.

[2] Carterette, B., Kanoulas, E., Hall, M. M., and Clough, P. D. Overview of the trec 2014 session track. In *Text Retrieval Conference* (2014).

[3] Jiang, F. Identifying and mitigating vulnerabilities in llm-integrated applications. Master's thesis, University of Washington, 2024.

[4] Liu, J., and Mozafari, B. Query rewriting via large language models. *arXiv preprint arXiv:2403.09060* (2024).

[5] Nguyen, T.-D., Bui, C. M., Vuong, T.-H.-Y., and Phan, X.-H. Passage-based BM25 hard negatives: A simple and effective negative sampling strategy for dense retrieval. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation* (2023), Association for Computational Linguistics, pp. 591–599.

[6] Reimers, N., and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (11 2019), Association for Computational Linguistics.

[7] Sinha, A. Don't retrieve, generate: Prompting llms for synthetic training data in dense retrieval. *arXiv preprint arXiv:2504.21015* (2025).

[8] Sun, Z., Zhou, X., and Li, G. R-bot: An llm-based query rewrite system. *arXiv preprint arXiv:2412.01661* (2024).

[9] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[10] Xiao, S., Liu, Z., Zhang, P., and Muennighoff, N. C-pack: Packaged resources to advance general chinese embedding, 2023.

[11] Yang, Z., Shao, Z., Dong, Y., and Tang, J. Trisampler: a better negative sampling principle for dense retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2024), vol. 38, pp. 9269–9277.

[12] Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., and Ma, S. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval* (2021), pp. 1503–1512.

A Rewrite Prompt and Example

```
SYS = Rewrite a user's search query to be more personalized, using the current query, topic,
user-clicked documents, and past session queries.
Consider the specific context and interests of the user based on their past interactions and
current session data. Use this information to tailor the query to better match the user's likely
intent and preferences.

# Steps
1. Analyze the Current Query: Understand the main intent and subject matter of the current search
query.
2. Review Past Queries: Examine the past session queries to identify patterns, recurring themes,
or topics of interest to the user.
3. Evaluate Clicked Documents: Look at the content of documents the user has clicked on for
additional context and preferred information types.
4. Determine User Interests: Synthesize information from past queries and clicked documents to
identify key interests and preferences.
5. Rewrite Query: Use insights gathered to create a more personalized query that better reflects
the user's likely interests and enhances relevance.

# Output Format
Return only the rewritten query. Do not include explanations, markdown, or formatting.
The personalized query should be concise and directly reflect the user's inferred intent and
preferences, with adjustments made to improve relevance and specificity.
```

```
Input:
{
     "query": "seattle vacation",
     "positive_doc": "clueweb12-0112wb-18-23823",
     "session_queries": ["hawaii vacation"],
     "topic_desc": "Suppose you'd like to take a week-long vacation.  Research some
     possible destinations.  How do they compare in terms of cost (travel + room and
     board + entertainment), value (things to do), and feasibility (ease of getting
     there)?"
}

Output: "Top destinations for a week-long vacation in the US: compare costs, values,
and feasibility of Hawaii, Florida, and California. Get exclusive deals and
discounts on Hawaii vacation packages, including all-inclusive options and luxury
rentals. Consider visiting Disney World in Florida or exploring the natural beauty
of California's coastline and national parks. Find the best value for your money and
make the most of your vacation time."
```

Figure 1: Long Query Rewrite Prompt and Example

```
SYS = Rewrite a user's search query to be more personalized, using the current query, topic,
user-clicked documents, and past session queries. Ensure the rewritten query consists of ten
words or fewer.
Consider the specific context and interests of the user based on their past interactions and
current session data. Use this information to tailor the query to better match the user's likely
intent and preferences.

# Steps
1. Analyze the Current Query: Understand the main intent and subject matter of the current search
query.
2. Review Past Queries: Examine the past session queries to identify patterns, recurring themes,
or topics of interest to the user.
3. Evaluate Clicked Documents: Look at the content of documents the user has clicked on for
additional context and preferred information types.
4. Determine User Interests: Synthesize information from past queries and clicked documents to
identify key interests and preferences.
5. Rewrite Query: Use insights gathered to create a more personalized query that better reflects
the user's likely interests and enhances relevance.

# Output Format
- Return only the rewritten query. Do not include explanations, markdown, or formatting.
- The rewritten query must be ten words or fewer.
- The personalized query should be concise and directly reflect the user's inferred intent and
preferences, with adjustments made to improve relevance and specificity.
```

```
Input:
{
"query": "seattle vacation",
"positive_doc": "clueweb12-0112wb-18-23823",
"session_queries": ["hawaii vacation"],
"topic_desc": "Suppose you'd like to take a week-long vacation.  Research some
possible destinations.  How do they compare in terms of cost (travel + room and
board + entertainment), value (things to do), and feasibility (ease of getting
there)?"
}

Output: "Best vacation spots for a week-long trip: Hawaii vs. Florida vs. Canada.
Compare costs, value, and ease of travel."
```

Figure 2: Short Query Rewrite Prompt and Example

```
SYS = Extract relevant keywords from the user's search context (topic, clicked documents, and
past queries).

# Rules
- Extract 2-3 most relevant keywords
- Keywords should be single words
- Keywords should reflect user interests and search intent
- Return keywords separated by comma

# Output
Only return the keywords separated by comma. No explanations.


**Instructions**:
- Extract 2-3 most relevant keywords from the context
- Return only the keywords separated by comma
**Keywords**:"""
```

```
Input:
{
"query": "seattle vacation",
"positive_doc": "clueweb12-0112wb-18-23823",
"session_queries": ["hawaii vacation"],
"topic_desc": "Suppose you'd like to take a week-long vacation.  Research some
possible destinations.  How do they compare in terms of cost (travel + room and
board + entertainment), value (things to do), and feasibility (ease of getting
there)?"
}

Output: "seattle vacation hawaii travel"
```

Figure 3: Query Keyword Rewrite Prompt and Example