Information Retrieval and Recommender Systems a.y. 2024 - 2025

Emirhan Kayar Nicolò Cappa Alessandro Longato

February 5, 2025

Personalized Information Retrieval

1 Introduction

This project aims to develop a personalized search engine that retrieves relevant answers from the SE-PQA dataset, a community question answering resource (CQA). The key innovation lies in integrating personalization into the search process, tailoring results based on user-specific features such as contextual data and preferences.

The project combines foundational and advanced methods from **personalized information retrieval** (PIR). Core techniques include traditional statistical retrieval models like **TF-IDF** and **BM25**, complemented by neural re-rankers such as **bi-encoders** and **cross-encoders**. The system will also leverage user-level features via **Large Language Models** (LLMs), by expanding queries to enhance the searching experience.

In addition, **methods inspired by recommender systems** are applied to improve personalization. User profiles are built based on tags associated with their historical activity, and content relevance is calculated using similarity measures. By combining these recommender-driven techniques with robust search functionality, the system aims to deliver a tailored and efficient search experience.

To note, we did not perform a grid search to find optimal weighting combinations for each test due to hardware and time constraints. Instead, we manually selected reasonable configurations as a demonstration. This aspect could be further explored in future work.

2 Our Idea

The goal is to design a **fully transparent and customizable search engine**. Given a user profile, a natural language query, and the chosen retrieval method, the system processes the input and returns the top 10 most relevant documents from the corpus based on the selected technique. Users have complete control over the retrieval methods employed, encouraging trust and empowering them to tailor their experience. This approach gives users both the responsibility and the capability to shape their interaction with the system according to their preferences. We tested and developed different retrieval methodologies to find the most optimal ones. The user then may choose which of those to employ.

3 Dataset

We utilized the SE-PQA dataset [1], a comprehensive resource containing over one million questions and over two million answers extracted from the StackExchange forums. This dataset includes rich metadata, making it particularly valuable for leveraging user-level and social-level features in information retrieval tasks.

For our experiments, we used a subset comprising slightly fewer than 10,000 documents (answers) as the corpus for the retrieval task. We used 100 queries to train, 100 queries to validate, and 100 queries to test our retrieval system.

In addition, we incorporated metadata related to questions and answers - specifically tags for both, as well as scores and comment counts for the answers - to refine and expand the retrieval pipeline.

4 Methodologies

4.1 Preprocessing

Firstly, the corpus and the queries were preprocessed by converting all words to lowercase and removing punctuation marks. Two different versions of the corpus were then created: one underwent stemming and stop-word removal, while the other was limited to basic preprocessing. We evaluated both versions using our baseline models to determine which to retain for subsequent analysis. The results indicated that basic preprocessing was preferable. This decision was based on the observation that stemming did not significantly reduce the vocabulary size and that neural models, such as BERT, tend to perform better on minimally processed corpora, as suggested in the original paper [2]. This is because BERT captures semantic relationships more effectively when full words are preserved.

4.2 Baseline

First, we designed two baseline models to use as reference for our experiments. We started from the traditional statistical models of information retrieval: **TF-IDF** and **BM25** [3, 4].

The TF-IDF model uses a statistical measure to compute the relevance of each document in relation to the query. It is formed by two weights. The term frequency $\frac{tf_{t,d}}{\max_{t_i \in d} tf_{t_i,d}}$ represents the term occurrences within a document, while the inverse document frequency $idf_t = \log(\frac{N}{df_t})$ represents the rarity of a term in the collection [3].

The BM25 model can be seen as a refinement of the TF-IDF: $\text{BM25} = idf_t \cdot \frac{tf_{t,d} \cdot (k_1+1)}{tf_{t,d}+k_1 \cdot (1-b+b \cdot \frac{|d|}{\text{avgdl}})}$. The IDF is computed as before, while it adds two important improvements to the TF. The first one is term saturation, as a cap on the influence of term frequency is introduced using k_1 (typically 1.2), and the second one is document length normalization based on the ratio between the document length and the average document length in the corpus, using b (typically 0.75) [4].

The relevance score of a document d related to a query q is then computed as $\sum_{t \in q} \text{TF-IDF}(t, d)$ or $\sum_{t \in q} \text{BM25}(t, d)$, respectively.

4.3 Neural Re-rankers

We then implemented neural re-rankers to improve the initial ranking produced by simpler models. Neural-based re-ranking techniques were applied to the top 100 most relevant documents. We used two types of Hugging Face BERT-based model: **bi-encoder** [6] and **cross-encoder** [7] . Both models were pre-trained on the MS MARCO (Microsoft MAchine Reading COmprehension) dataset [5]. These models are based on the **Sentence-BERT architecture**. Although these models were not explicitly trained on question-and-answer data, they could be fine-tuned using the training data from the SE-PQA dataset. However, due to satisfactory performance and limitations in time and resources, we chose not to perform fine-tuning. This leaves room for future work, as training data can be created using the standard approach of labeling one positive and one negative example per query-document pair for fine-tuning.

The bi-encoder processes the query and document separately, producing independent embeddings for each. The similarity between the two embeddings is then computed to determine relevance. To ensure efficient online retrieval, we precomputed the embeddings for all documents in advance.

In contrast, the cross-encoder jointly encodes the query and document pair, modeling the interaction between the two inputs during the encoding process. Although this approach yields more accurate and contextually relevant results, it is significantly more computationally expensive. For this reason, the cross-encoder was primarily employed during the reranking phase. A scoring function was applied to the encoded inputs to determine the final ranking scores.

4.4 LLM

The implementation of **Large Language Models** (LLMs) is a critical part of phase II development, aimed at enhancing the retrieval and recommendation pipeline through personalized query expansion. For this purpose, the "microsoft/Phi-3-mini-4k-instruct" model, a pre-trained LLM designed specifically for instruction-following tasks [8], is utilized to generate contextually enriched and personalized queries based on user-level data.

Despite its promise, the performance of the expanded queries has been observed to be lower than that of the original queries. One primary issue is the tendency of the LLM to significantly elongate queries when incorporating user data. While the expanded queries remain contextually relevant, their increased length can dilute the focus of the retrieval system, leading to decreased performance.

A common challenge with LLMs is the issue of "**hallucination**", where the model generates plausible-looking information that is factually irrelevant. This can introduce noise into the query expansion process, adversely affecting

retrieval precision. Additionally, the "black-box" nature of LLMs makes it difficult to interpret why certain expansions are generated and how they influence the final retrieval outcomes.

Another observed issue is the handling of incomplete queries present in the data. When the LLM attempts to expand or complete such queries, it can lead to "**over-expansion**", introducing unnecessary or irrelevant details that distort the original intent of the query. Although this completion may occasionally prove beneficial, it often reduces overall performance by diverging from the core semantics of the original query.

Furthermore, the approach of embedding historical tags directly into the query text may not be the most effective way to incorporate user-specific information. Since the expanded query must still pass through the retrieval system before influencing the ranking score, this method does not fully leverage the personalization potential. To address this limitation, alternative techniques were explored, focusing on combining features from both personalized and non-personalized recommender systems. These methods aim to directly influence the final ranking score by integrating user-level and content-based features in a more effective manner.

4.5 Personalization

The richness of the dataset enabled us to experiment with recommendations for improving retrieval and user experience. We began with an approach inspired by **non-personalized** recommender systems, using the 'Score' of the 'answer' table as a proxy for answer quality. For each document retrieved, the system fetches its score, normalizes it to a [0, 1] range, and uses it for weighted reranking. Similarly, 'CommentCount' serves as a proxy for answer popularity, with normalized values incorporated into the reranking process.

The personalization model in the dataset paper mirrors collaborative filtering, considering the similarity between tags in the querying user's questions and the answering user's answers [1]. Inspired by **content-based** techniques, we instead assessed document relevance using features tied to querying user preferences. Specifically, the querying user's profile is built from tags linked to their authored questions and answers. A binary vectorizer is trained on these tags, transforming them into a binary vector. Each document's tags are similarly transformed, and the cosine similarity between the user's vector and the document vector is computed. To address low support in document vectors, the similarity score is normalized by the minimum magnitude of the two vectors, enhancing its significance in weighted reranking.

We integrated this system into the **PyTerrier** pipeline with custom transformers, each computing specific scores and appending them as columns to the dataframe. This allows seamless incorporation of additional features and efficient calculation of weighted averages for final rankings, maintaining pipeline compatibility.

4.6 Final Search Pipeline

After determining the best configurations for each individual system, we combined them into a final function. This function takes as input the user ID, query text, and relevant function arguments, and returns the top 10 documents retrieved using the chosen method. The process begins by building a user-specific dataframe that incorporates historical data. Following this, the query is processed and the search is executed, returning the top 10 documents along with the total processing time. This processing time serves as a metric for users to evaluate the overall efficiency of the selected method, providing transparency and enabling informed decision-making.

5 Evaluation

We used the same metrics throughout the project to ensure consistency and comparability of the results. The metrics we chose are designed to provide a comprehensive evaluation of the retrieval system's performance across different aspects:

- **R@100 and R@5:** Recall metrics assess the system's ability to retrieve relevant documents, with R@5 focusing on relevance in the top-ranked results.
- **P@1:** Measures whether the top-ranked document is relevant, crucial for ensuring high-quality first results.
- **AP@100:** Provides a single score summarizing ranking quality and retrieval effectiveness across recall levels.
- **nDCG@10 and nDCG@3:** Evaluate ranking quality by giving more weight to relevant documents at higher ranks, reflecting user satisfaction with top results.

By using these metrics, we ensure that our evaluation captures both the system's ability to retrieve relevant documents (recall-focused metrics) and its effectiveness at ranking the most relevant documents at the top (precision and nDCG).

This comprehensive approach aligns with the goals of personalized search, where both relevance and ranking quality significantly impact user experience.

Model	**R@100**	**R@5**	**P@1**	**AP@100**	**nDCG@10**	**nDCG@3**	λ
BM25	0.928571	0.826531	0.714286	0.763667	0.789195	0.760446	-
BM25 + BI + CROSS	0.928571	0.897959	0.744898	0.808658	0.832735	0.816811	(.2;.4;.4)
BM25 + BI + CROSS + QUALITY	0.928571	0.908163	0.775510	0.822925	0.846119	0.817905	(.1;.4;.4;.1)
BM25 + BI + CROSS + POPULARITY	0.928571	0.918367	0.775510	0.823258	0.846431	0.817905	(.1;.4;.4;.1)
BM25 + BI + CROSS + CONTENT	0.928571	0.908163	0.775510	0.823443	0.846566	0.817905	(.1;.4;.4;.1)

Table 1: Performance metrics for the **final test**.

The evaluation results highlight the performance improvements achieved by incorporating additional features into the retrieval pipeline.

- **Baseline Performance (BM25):** The BM25 model shows robust recall metrics, with **R@100** at 0.928571 and **R@5** at 0.826531. However, its **P@1** is lower at 0.714286, indicating that while BM25 retrieves a broad set of relevant documents, its ranking of top results needs refinement. Similarly, **nDCG@10** is 0.789195, suggesting room for improvement in the ranking quality.
- **BM25 + BI + CROSS:** Adding the neural re-rankers (bi-encoder and cross-encoder) keeps **R@100** identical to BM25. However, there is a significant improvement of **R@5** at 0.897959, suggesting that the neural rerankers do a better job at recalling the top most relevant documents. There is also a slight improvement in precision metrics (**P@1** = 0.744898 and **AP@100** = 0.808658), alongside a significant improvement of the ranking metrics both at **nDCG@10** and **nDCG@3**. This shows that **neural re-rankers do a good job of ranking the most relevant documents**.
- **Quality Feature (BM25 + BI + CROSS + QUALITY):** Introducing a **quality** score boosts **R@5** to 0.908163, **P@1** to 0.775510, and **nDCG@10** to 0.846119, improving slightly over the neural ranker configuration. These improvements highlight the value of quality-based re-ranking, which prioritizes more intrinsically valuable results, thereby enhancing the precision and relevance of the top-ranked items.
- **Popularity Feature (BM25 + BI + CROSS + POPULARITY):** Adding the **popularity** score further enhances **R@5** to 0.918367. This suggests that integrating popularity metrics—such as user engagement—improves the ranking of more relevant and highly engaged items, demonstrating the system's sensitivity to users behavior and preferences.
- **Content-Based Personalization (BM25 + BI + CROSS + CONTENT):** Incorporating **content-based personalization achieves the highest nDCG@10** (0.846566), signifying the system's capacity to tailor results based on user-specific content preferences. This emphasizes the importance of personalization for achieving the best ranking quality.

6 Conclusion

In conclusion, the project was **overall a success**. We successfully built a search engine with satisfactory performance, effectively integrating personalization data. The final pipeline demonstrates how, given two different users with distinct profiles, the system can re-rank the set of retrieved documents to fit the context of each user.

The project presented several **challenges**, such as gaining a deep understanding of the **PyTerrier framework** and managing **hardware constraints**. Many tests required hours to complete, which forced us to work in parallel and efficiently utilize the results of one session to proceed with the next steps. Despite these obstacles, we successfully managed the workload by distributing tasks equally among team members.

We are proud of the final results; however, there are still many **improvements that could be explored in the future**. For instance, we could fine-tune the models, we could perform a grid search to find optimal weight combinations, test the system on the full dataset rather than a subset, optimize performance of the content base weight computations, experiment with combining the three different recommendation scores simultaneously, or try larger models to further enhance the system's capabilities.

References

[1] P. Kasela, M. Braga, G. Pasi, and R. Perego, *SE-PQA: Personalized Community Question Answering*, arXiv preprint arXiv:2306.16261, 2023. Available at: `https://doi.org/10.48550/arXiv.2306.16261`.

[2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 4171–4186, 2019. Available at: `https://arxiv.org/abs/1810.04805`

[3] K. Sparck Jones, *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, Journal of Documentation, 28(1): 11–21, 1972.

[4] S. Robertson, S. Walker, M. Beaulieu, M. Gatford, and A. Payne, *Okapi at TREC-4*, In Proceedings of the Fourth Text REtrieval Conference (TREC-4), pp. 73–96, 1996.

[5] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, *MS MARCO: A Human Generated MAchine Reading Comprehension Dataset*, In Proceedings of the Workshop on Cognitive Computing at NIPS, 2016. Available at: `https://microsoft.github.io/msmarco/`

[6] N. Reimers and I. Gurevych, *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*, In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019. Available at: `http://arxiv.org/abs/1908.10084`

[7] Hugging Face, *ms-marco-MiniLM-L-6-v2*, Cross-Encoder pre-trained on the MS MARCO dataset for passage re-ranking. Available at: `https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2`, accessed January 28, 2025.

[8] Hugging Face, *Phi-3-mini-4k-instruct*, Large Language Model by Microsoft for instruction-following tasks. Available at: `https://huggingface.co/microsoft/Phi-3-mini-4k-instruct`, accessed January 29, 2025.