Personalized Information Retrieval

SE-PQA DATASET

Barbieri Chiara 517096, Sotgia Francesca 513067

509496 - Information Retrieval and Recommender Systems

https://drive.google.com/file/d/1vTtAOZ6vzW_y8WybmRPPupkg8gb43WIB/view?usp=sharing

Introduction

The project focuses on developing a search engine tailored for a community Question Answering dataset (SE-PQA dataset). The main goal is to implement retrieval methods, such as TF-IDF and BM25, with neural re-rankers and personalized models to evaluate how the model performs before and after adding advanced recommender systems' features to the traditional information retrieval methods.
The dataset contains questions, answers, user ids, scores and metadata, such as tags and views, which provide the needed information to develop a personalized search engine tailored to the users' activity.
The results suggest that personalization improves the generalizability and the accuracy of the model at predicting relevant answers for a user's question.

Intuition of the idea

The main idea of this project is to combine the strengths of probabilistic IR models with neural re-ranking techniques and personalized models to retrieve relevant answers for a given question, based on the users' activity.
Traditional models are efficient at ranking relevant answers for a given question, but to deeply understand the relationship between queries and documents we need to add a neural re-ranker as Cross Encoder. Furthermore, by integrating query expansion and user-based personalization, we expect the model to be able to better generalize and to have improved performance.

Methodology

We started by downloading PyTerrier and Retriv libraries, which are suitable for information retrieval tasks and that will be used throughout the project.

Dataset downloading and preprocessing

We proceeded by downloading the dataset and exploring its content to understand the structure, in order to identify the relevant files to use in the project.
We then preprocessed the data, which is a crucial step because it transforms the data into a more meaningful format, and after applying the preprocessing to the dataframes we converted them into a Retriv-friendly format (list of dictionaries).

Model selection

We defined and applied to the preprocessed corpus two SparseRetrievers (from the Retriv library), one with BM25 model and the other with TF-IDF, in order to index the documents and compare the performance of the models.
To evaluate them we used the metrics MAP@100, F1 and P@5 and P@10; the results showed that the best model for our task is BM25 (MAP@100 for BM25 is 0.721 and

MAP@100 for TF-IDF is 0.380) and thus it's the one to be applied to have better performances.

Neural re-ranking

To understand more in depth the relationships between the queries and the documents, we implemented a Cross Encoder.
Cross Encoder is ideal to verify question-answer pairs and to entail the sentences, it ideally makes the model more suitable for a search engine also due to its contextual understanding.

Since SparseRetriever does not provide a valid index for PyTerrier (which is used for the Cross Encoder), we indexed the documents using the PyTerrier function "*IndexFactory*".

After indexing, we defined the normalized pipelines for the BatchRetrieve and the Cross Encoder, using different combinations of the "*normalized_cross_pipeline*" and "*normalized_br*".
We then ran an experiment on the validation set (we used a subset to deal with computation complexity) to assess which pipeline is the best one for our data.
Once we compared the scores and identified the best pipeline, we ran the experiment on the test set and we found that the identified pipeline is suitable for the test set and returned better scores compared to the ones for the validation subset.

Query expansion

Query expansion can be implemented into the model to refine the questions and to improve the efficiency of the model. Moreover, after the expansion we expect better generalization of the model on unseen queries and a better understanding of the context by the search engine. This could result in better query-answer matching and, thus, better evaluations.

To implement query expansion we introduced a pre-trained LLM model ("*microsoft/Phi-3-mini-4k-instruct*") that will expand the queries of the dataset using the users' data and context.
We defined and applied the functions for the query expansion and updated the dataset with a new column "*query_expanded*" which is saved into the drive with the name "*dataset_with_expandedQueries*" for efficiency purposes.

We proceeded with transforming the items to strings and preprocessing the expanded queries with the defined functions. Moreover, we splitted the "*dataset_with_expandedQueries*" into train and test (respectively 80% and 20%) and transformed the queries into a Retriv-friendly format (list of dictionaries containing users' ids and expanded queries) and finally, we defined the qrels for both sets.

We then ran an experiment on the train set to identify the best pipeline to be used on the train set. We noticed that the results are worse than the ones obtained with the Cross Encoder and we expect better performance once the model is personalized.

Personalized model

To add user-based personalization to the model we implemented Content Based Filtering (CBF) and User-based Collaborative Filtering (CF).
Content based filtering suggests relevant answers for a user based on the characteristics of the answers and the user's context to ensure more personalized recommendations.
User-based collaborative filtering focuses on the interactions between users and items to suggest answers based on what similar users interacted with.
By combining these methods we expect recommendations tailored to the individual preferences and the collective behavior and thus a higher performance of the model.

We started by defining a matrix containing the retrieval scores for each query. Then we calculated the CBF scores and matrix to compute the similarities between the queries, using cosine similarity, and the CF scores and matrix to compute the similarities between the users, also using cosine similarity.
Moreover, we combined the retrieved scores with the CBF and CF scores to obtain the personalizations scores.
After extracting the personalized qrels, which are the most relevant documents based on the personalization scores, we ran an experiment on the train set to identify the best pipeline to be used on the train set and to evaluate the model's performance.
By analyzing the results we can conclude ….

Experimental setup

The SE-PQA dataset, stored into pir_data.zip, consists of multiple CSV and JSONL files containing information about the users, questions, answers and metadata (such as tags and comments) useful to retrieve context and interactions. Furthermore, the dataset is divided into train, validation and test subdirectories and each one contains two files, one collecting questions and answers (*subset_data.jsonl*) and the other collecting the relevance of the answers for the questions (*qrels.json*).

Data preprocessing

Since the data might contain missing values or noisy data, we defined the preprocessing functions to be applied to the queries and the answers of the train, validation and test dataframes to have the text written in a consistent way.
The functions are: "*preprocess_text_basics*", which consists of lowercasing, remove numbers, links and extra spaces present in the text and "*preprocess_text_norm*", which includes tokenization, stop word removal and stemming.

After applying the preprocessing function to the dataframes, we converted them into lists of dictionaries containing user's id and the text to be compatible with Retriv.

Evaluation metrics

In the project we used multiple evaluation metrics, such as MAP, Precision, Recall, nDCG and F1. We chose to consider Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (nDCG) as the primary evaluation metrics in the project and they were used to evaluate the performance of the model, which is how good it was at predicting the best answer for a given query.

Experimental results

Using traditional IR models with Cross Encoder we obtained the baseline scores MAP=82.75% and nDCG=85.68% which show high performance of the model.
The scores obtained on the dataset with expanded queries are lower than the baseline (MAP=57.93% and nDCG=63.81%), probably due to the lack of appropriate scores, so we expect a better performance once the personalized model is implemented.
The final model, obtained after the implementation of the personalized model and the recommender system, performs worse than the model with only the Cross Encoder, this can happens for a variety of reasons discussed in the conclusion.

Model	**MAP**	**nDCG**
Baseline	0.8275	0.8568
Expanded queries	0.5793	0.6381
Personalized	0.4461	0.5306

Conclusion

In this project we analyzed how advanced techniques such as neural re-rankers, query expansion, personalized models and recommender systems impact on the performance of a search engine applied to a community Question Answering dataset. In contrast with our expectations, the personalized model has a lower efficacy at retrieving relevant answers compared to the baseline model. The outcome can be attributed to multiple factors such as the introduction of noise by the LLM during query expansion and the potential introduction of biases during the personalization, caused by the limited information about the users.
Potential future works could focus on improving the results by refining the query expansion and the personalized models. This could be done by leveraging context-awareness and more advanced personalization techniques.
Overall, the project demonstrates the theoretical importance of advanced models while showing that their application depends on the task, dataset and user needs.