# Information retrieval and Text Mining : Group Project

Til Mohr
tv.mohr@stud.uis.no

Ishaac Ourahou
i.ourahou@stud.uis.no

## ABSTRACT

As part of our master's degree, we study information retrieval and text mining. This group project aims to design a system that addresses the problem of searching for passages within a conversation. This therefore consists of taking the context of the previous conversation history into account of the current user query, rewriting the current user query in a way that encompasses the conversation topic, selecting the most relevant passages to that rewritten query, through a collection of 8.8 million documents (MS MARCO passages). Based on the knowledge acquired on the subject, we present a baseline information retrieval pipeline, which will serve as our reference system for future comparisions with more advanced information retrieval pipelines. Afterwards, we dive into more advanced approaches in this field of conversational search engines. Finally, we make comparisons between the baseline system and the implemented advanced retrieval pipelines to highlight the positives and negatives of our implementations, and also draw conclusions.

## KEYWORDS

information retrieval, conversational search engine, conversational search system

## 1 INTRODUCTION

Nowadays, we can see a strong development of conversational agents like Amazon Alexa or Apple Siri. These conversational agents aim to communicate with a user and return relevant responses to the requests that the user formulates. The need to return the right information is important to ensure the quality of the responses sent to the user.

The goal of our project is to create a conversational search engine, i.e. a system where the user may ask multiple questions (queries), and the system takes the conversation history, e.g. the current topic of the conversation, as well as the current query into accout to determine the most relevant documents to the user. We will design a simple basic architecture which will serve as our baseline system for future comparision with more advanced approaches. However, as a hard requirement, this baseline retrieval pipeline must already perform at least as well as the reference system provided by our project owners.

For this project, we will use the MS MARCO dataset, which contains 8.8 million documents. This document collection has to be indexed in order to be able to retrieve relevant documents efficiently. This indexing will be achieved through the `pyterrier` API.

In section 2 we will define the problem statement at hand. Following the problem statement, we will dive into the implementation of the different retrieval pipelines we have developed. In general, retrieval pipelines can be segmented into the following stages: Query rewriting, (multi-pass) retrieval, reranking. In sections 3 and 4, we will present our baseline and advanced retrieval pipelines in aspects of these components, respectively. Finally, in section 5, we will present the performance results of our retrieval pipelines and compare them to each other, highlighting any improvements and drawbacks of our advanced retrieval pipelines.

## 2 PROBLEM STATEMENT

The goal of this project is to design a conversational search engine (CSE). The main differeation of such a system to a ordinary search engine is that queries asked by the same user within a session are not independent of each other in a CSE. A CSE takes currently and past discussed topics into account for searching relevant passages. The system should be able to respond to queries which continue the flow of the conversation, as well as handle abrupt changes in the conversation topic. For each query, the system should then return a ranked list of the top 1000 most relevant passages from a collection of 8.8 million documents.

In addition to the components of a ordinary search engine retrieval pipeline, a CSE retrieval pipeline should also include a query rewriting component that rewrites the current query to reflect the current conversation topic. A overview of how the different components of a CSE retrieval pipeline interact with each other is shown in figure 1. As it is common practice in ordinary search engines, also in CSE the document collection must be indexed beforehand to facilitate efficient document retrieval and ranking.

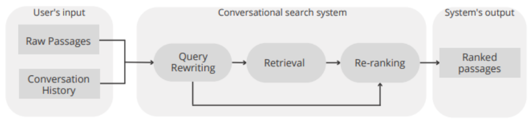

**Figure 1: Global Pipeline architecture of the system**

## 3 BASELINE METHOD

Explain what you are taking as your baseline method, as well as why this is a reasonable baseline, and why you are making specific implementation choices.

Our baseline method is inspired by the baseline method presented by Łajewska et al. [1]. Our baseline method consists of the following stages:

(1) `T5` Query Rewriting
(2) `BM25` Retrieval
(3) Re-ranking
  (a) Re-ranking with `monoT5`
  (b) Re-ranking top documents with `duoT5`

### 3.1 T5 Query Rewriting

In conversation search engines, query rewriting is the crucial component to include the semantics of the conversation history into the currently asked query, which results into a singular rewritten query that can be passed along into the retrieval pipeline.

For this purpose, we include all the previously rewritten queries $q'_0 \dots q'_{n-1}$ of our conversation, as well as the response $r_{n-1}$ of the CSE to the previous rewritten query $q'_{n-1}$ into the current query $q_n$. This is done by concatenating the previous rewritten queries and the response into a single string:

$$q'_n \coloneqq q'_0\text{<sep>}\dots\text{<sep>}q'_{n-1}\text{<sep>}r_{n-1}\text{<sep>}q_n$$

This approach resembles the approach taken by Łajewska et al. [1].

We have found through experimentation that this concatenation is insufficient to produce satisfiable results: The concatenated string is too long, and too much focus during the later retrieval is being put on $r_{n-1}$, which make sudden changes in topic impossible.

For those reasons, we have turned our attention to other query rewriting methods, which we could append to the currently followed approach. `Pyterrier` provides the `SequentialDependence` query rewriting method[1]. We have found, however, that this rewriter does not produce the desired result.

Researching further, we stumbled upon the `T5` neural query rewriter trained on conversational question rewriting[2]. This method produced the desired results: The rewritten query $q'_n$ resembles the original query $q_n$, just rewritten very slightly to include the conversation topic, if no sudden change was detected. As a side effect, the rewritten query is now also very small in size, and not growing with the number of queries asked in the conversation, as we saw in the previous approach.

### 3.2 BM25 Retrieval

The `BM25` retrieval method is a common retrieval method in information retrieval. We chose to use this retrieval method, as the reference system also uses this basic retrieval method. This allows us to easily compare our baseline system to the reference system.

### 3.3 Re-ranking

The re-ranking stage of our baseline system consists of two stages: First, the top 1000 documents retrieved by the `BM25` retrieval method are re-ranked using the `monoT5` reranker. Afterwards, the top 50 documents of the previous re-ranking stage are re-ranked using the `duoT5` reranker. The actual number of documents to be reranked in each stage is a hyperparameter of our system, and can be adjusted to trade off runtime and quality of the result. The rerankers were implemented in the `pyterrier_t5` library.[3]

## 4 ADVANCED METHOD

Explain what you are taking as your advanced method(s), as well as why this is a promising attempt to outperform the baseline method, and why you are making specific implementation choices.

## 5 RESULTS

The individual methods were evaluated on the MS MARCO document collection and the following provided files: `queries_train.csv` holds a list of queries grouped together into several conversation sessions, and `qrels_train.txt` contains the relevance assessments for the training queries. Each method was then evaluated on the following metrics:

- Recall at 1000 (R@1000)
- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR)
- Normalized Discounted Cumulative Gain (NDCG_Cut@3)

These metrics were calculated using the `trec_eval` tool:

```
trec_eval -c -m recall.1000 -m map -m recip_rank
-m ndcg_cut.3 -l2 -M1000 qrels_train.txt
{GENERATED_TREC_RUNFILE}
```

The baseline method can be parameterized in a few different ways. For the evaluation, the following parameterization was used: The document retrieval (`BM25`) used the default `pyterrier` parameters[4] to retrieve the 1000 most-relevant documents for each query. All 1000 documents were then reranked using the `monoT5` reranker. Because of the high computational cost of the `duoT5` reranker, only of those 1000 documents the best 50 documents were then reordered using this reranker.

**Table 1: Performance of the different methods on the MS MARCO document collection.**

| | R@1000 | MAP | MRR | NDCG_Cut@3 |
|---|---|---|---|---|
| Reference System | ??? | ??? | ??? | ??? |
| Baseline Method | 0.1746 | 0.3230 | 0.5705 | 0.2461 |

## 6 DISCUSSION AND CONCLUSIONS

Summarize and discuss different challenges you faced and how you solved those. Include interpretations of the key facts and trends you observed and pointed out in the Results section. Which method performed best, and why? Speculate: What could you have done differently, and what consequences would that have had?

## REFERENCES

[1] Weronika Łajewska and Krisztian Balog. 2023. From Baseline to Top Performer: A Reproducibility Study of Approaches at the TREC 2021 Conversational Assistance Track. In *European Conference on Information Retrieval (ECIR '23)*. Springer, 177–191. https://doi.org/10.1007/978-3-031-28241-6_12

[1]URL: https://pyterrier.readthedocs.io/en/latest/rewrite.html#sequentialdependence
[2]URL: https://huggingface.co/castorini/t5-base-canard
[3]URL: https://github.com/terrierteam/pyterrier_t5
[4]URL: https://pyterrier.readthedocs.io/en/latest/terrier-retrieval.html

## A DIVISION OF WORK DURING THE PROJECT