Assembling four Open Web Search Components

TU Dresden at WOWS 2024

Linda Erben[1,†], Maria Hampel[1,†], Malte-Christian Kuns[1,†], Vincent Melisch[1,†], Per Natzschka[1,†], Wilhelm Pertsch[1,†], Lina Razouk[1,†], Reiner Stolle[1,†], Robert Thomas Thoss[1,†], Tuan Giang Trinh[1,†], Julius Gonsior[2,*] and Anja Reusch[2,*]

[1] *Technische Universität Dresden, Dresden, Germany*

[2] *Technische Universität Dresden, Dresden Database Research Group, Dresden, Germany, {first}.{last}@tu-dresden.de*

Abstract

In this work, we present the submission of TU Dresden to WOWS 2024. Four student teams assembled different approaches for Genre Classification, Text Snippet Extraction, Query Expansion, and Text Features. Each implemented component integrates seamlessly into the open web search ecosystem. We present each approach alongside a short evaluation of possible use cases, and hope that our submission will contain viable building blocks for future research to be build on top.

Keywords

Information Retrieval, Open Web Search, Genre Classification, Text Snippet Extraction, Query Expansion, Text Features

1. Introduction

This report describes the submission of the team at TU Dresden for the Workshop on Open Web Search WOWS 2024 [1]. The work was conducted during a university-organized hackathon targeted at students. Details about the setup are included in the Appendix in Sec. A. Four teams, consisting of two to three students contributed four components for the open web search ecosystem. We hope that with our submitted components future research on Information Retrieval (IR) can be facilitated.

In summary, this paper is discusses the following four components: Sec. 2 reports the work of the group Genre Classification, which categorizes web pages based on the intent of the page, such as Discussion or Shopping. In Sec. 3 we detail our the submission for the extraction of text snippets. Here, the goal is to divide long documents into shorter ones and return a list of the best snippets. Sec. 4 provides details on the work of the group Query Expansion, which employed Large Language Modelss (LLMs) to generate more related information or variants for a given query. The results for the extraction of text features is highlighted in Sec. 5. The goal of this component was to quantify syntactic or semantic features of natural language such as the readability of a web page. Finally, Sec. 6 draws the conclusions of all our submissions.

WOWS'24: 1st International Workshop on Open Web Search, March 28, 2024, Glasgow, Scotland

[*] Supervision of projects. Corresponding authors.

[†] Equal contribution in the respective components. These authors are ordered alphabetically after their last name.

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3689/WOWS_2024_paper_8.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

2. Genre Classificaion

The goal of genre classification [2] is to categorize documents into the intent of the document itself. Objectives for a website could include: (1) making sales (like in an online store), (2) providing information (like in a course website for a university), (3) sharing personal experiences (like in a personal blog), etc. Genre classification facilitates effective document filtering in ranking based on the search query in conjunction with existing query intent classifiers that differentiate search queries as *informational* (e.g. "What is IR?"), *transactional* (e.g., "I want to buy a PlayStation"), or *navigational* (e.g., "take me to log in for my university course") [3]. Relevant websites with a matching genre should be ranked higher if a query indicates an intent. This open web search component [1] examines techniques for classifying text documents into their respective categories, employing rule-based and machine-learning methodologies. We compare three classification strategies with a focus on high precision.

2.1. Methods

2.1.1. Rule-Based Classifier

The rule-based classifier makes use of a vocabulary list of relevant terms per genre. Comparing the intersection between terms in the genre-specific vocabulary lists, and the terms in the document, the most probable category is the one with the highest intersection. We first remove stop words and subsequently extract the 75 most frequent terms that we compare to the vocabulary lists to classify the genre. We use Snorkel AI [4] for implementation.

The rule-based classifier can be adapted to a precision-oriented method, where the most probable genre needs to be better than a threshold compared to the second most probable genre, otherwise the classification result is *abstain*.

2.1.2. Multi-Layer Perceptron Classifier

As a typical Machine Learning based method a neural network was used for classification. As features the web pages were converted into a tf-idf vectorspace. We use the Python library scikit-learn [5] for the implementation of the Multi-Layer Perceptron classifier. After an empirical hyperparameter search a neural network using a single hidden layer of 50 neurons, ReLU activation function, stochastic gradient descent in the Adam variant using momentum for optimizations, and a constant learning rate of 0.001 was used.

2.2. Experiments

2.2.1. Dataset

For evaluation we used the Genre-KI-04 dataset [2]. This includes vocabulary lists, and the following classification categories: articles, discussion, download, help, link lists, portrait (non private), portrait (private), and shop. Details about the genres can be found in the original paper.

[1]https://github.com/tira-io/workshop-on-open-web-search-tu-dresden-01

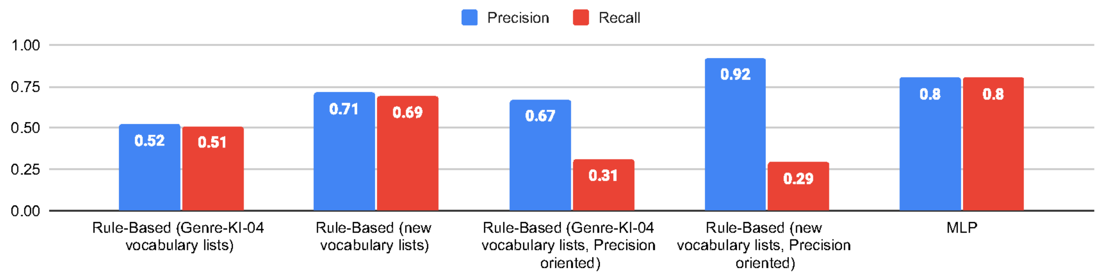


Figure 1: Precision and recall results for all methods

The dataset contains a total of 1,239 web documents classified into eight genres. The distribution of the classes is as follows: Articles: 127 (10.2%), Discussion: 127 (10.3%), Download: 152 (12.3%), Help: 140 (11.3%), Link lists: 208 (16.8%), Portrait (nonprivate): 179 (14.4%), Portrait (private): 131 (10.6%), and Shop: 175 (14.1%). The dataset is split into a training set with 928 documents and a test set with 311 documents. The corpus contains the HTML documents grouped into directories according to their respective genre. The first lines of each document contain the meta information for each document in an HTML comment. This information includes the URL from which the document was downloaded, the title, and the parsed text.

2.2.2. Vocabulary Lists

The rule-based classifier needs vocabulary lists containing core terms per genre. The Genre-KI-04 dataset contains a manually curated vocabulary list. Additionally we constructed an additional vocabulary list using the train set: First, all stop-words are removed. Secondly, the union of words shared among all genres is removed. Lastly, we performed a manual check over the vocabulary list. The remaining words are distinctive for the respective genres.

2.2.3. Results

Figure 1 shows the results of our compared approaches. We compared the rule-based approach both with our constructed vocabulary, and the given vocabulary in the dataset. The new vocabulary works better than the original vocabulary in the dataset. If a high precision is required, our rule-based approach should be used, otherwise the neural network provides the best overall performance.

2.3. Summary

We implemented a genre classification component for the open web search framework. Using our implemented methods, documents can be classified based on their intent. If the query or user intent is detectable, the genre classification can be used to improve the retrieval performance by ranking genre-specific documents higher than non genre-specific documents. We proposed multiple classifiers, including a precision oriented rule-based classifier and a neural network showing the best overall performance.

3. Text Snippet Extraction

Since sophisticated neural ranking models such as cross-encoders generally require a lot of computational effort, a customary retrieval pipeline first retrieves a number of (e.g., 1000) documents using a fast but imprecise retrieval method and then re-ranks those documents using a more precise weighting model [6]. Cross-encoder as introduced by Nogueira and Cho [6] are an example for the latter, which are used to calculate scores for query-document pairs. Apart from their comparatively high computational cost, cross-encoders have another disadvantage–their limited input size. This weakness is typically mitigated by truncating the document once the maximum number of input tokens is reached. The problem of this procedure is that content which is not in the beginning of a document is not taken into account by cross-encoders. As a result, the ranking of documents may be biased towards those that address the query early on.

In this part, we therefore present a simple method of extracting a number of snippets, i.e., smaller chunks of the document which fit in the cross-encoder as an additional component in a larger retrieval pipeline. Instead of simply truncation documents after a fixed number of tokens, we search for the most relevant passages (*ranked snippets*) in the document. These ranked snippets are used for the cross-encoder with the goal of a more precise ranking. We show the benefits of this method on two exemplary datasets which contain long documents.

3.1. Methodology

The re-ranking process with ranked snippets consists of five steps. An example of those steps for the re-ranking of $n = 3$ documents (d_1, d_2, and d_3) is shown alongside the explanation.

First, we subdivide all n documents into snippets. The maximum length of those snippets may be chosen arbitrarily–we defaulted to 250 tokens which is the passage size used by Dalton et al. [7]. The actual length of the snippets may vary since the division process aims to retain context by not separating sentences. For example, we may start with three documents d_1, d_2, d_3. After the first step, each of these documents is divided into several snippets: $s_1^1 s_1^2 \dots s_1^{l_1}$, $s_2^1 s_2^2 \dots s_2^{l_2}$, $s_3^1 s_3^2 \dots s_3^{l_3}$ where s_i^j denotes the j-th snippet of document d_i for $j \in \{1, \dots, l_i\}$ and l_i is the number of snippets of d_i.

In Step 2, we pre-rank all extracted snippets in relation to the query. To accomplish this, we view the set of all snippets of a document as a corpus. From this corpus, we can create a ranking for the query using one of the following weighting models: Term frequency (TF), BM25 or PL2. We do not use cross-encoder for the pre-ranking of documents, because there may be a multitude of snippets per document depending on document length and therefore ranking all snippets using a cross-encoder can drastically slow down the re-ranking process. After this pre-ranking step, our example snippets might be ranked in the following way: $s_3^3 > s_2^2 > s_2^4 > s_3^2 > s_3^1 > s_1^3 > s_3^4 > s_1^1 > s_2^3 > s_1^5 > \dots$.

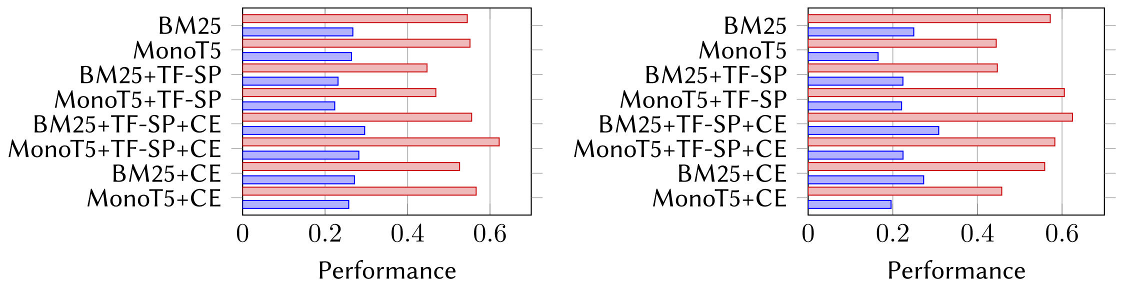
In Step 3, we can obtain the top k relevant snippets of each document from the pre-ranking, which are later ranked using a cross-encoder. This step ensures that the cross-encoder only needs to rank $n \cdot k$ snippets for n documents instead of all snippets. In order to reduce computational cost, we defaulted to $k = 3$. In our example, this step results in the following selection: $\{ s_1^3, s_1^1, s_1^5 \}$, $\{ s_2^2, s_2^4, s_2^3 \}$, $\{ s_3^3, s_3^2, s_3^1 \}$. Here, s_3^4 is not selected as one of the top snippets of d_3 since it is the $(k+1)$-th snippet of d_3 in the ranking despite being ranked relatively high.

In Step 4, the top k snippets of all documents are ranked using a cross-encoder (CE). That way, similar to Step 2, we can more accurately deduce which snippets best match the query–but now the ranking is more precise since we used a CE instead of the simple weighting models used in Step 2. An examplary ranking for our snippets might be: $s_2^4 > s_3^3 > s_2^2 > s_1^5 > \ldots$. The final document ranking ensues from this snippet ranking in Step 5, i.e., the document that provided the best snippet is ranked first. Our example documents are therefore ranked in the following way: $d_2 > d_3 > d_1$. It should be noted that the goal of this section is to rank documents with regard to a query, and not only passages. Therefore, the result is a ranking of documents. Details on our implementation can be found in Appendix B.1.

3.2. Evaluation

In this section, we conduct tests to study the possible improvements of our cross-encoder re-ranking of top k snippets. As baselines, we use BM25 and the dense retriever MonoT5. All further ranking is performed on the top 20 documents retrieved by these two systems. We evaluate the re-ranking with our TF-ranked snippets. For this, we load the previously saved top 3 snippets for each document. To re-rank the documents, we follow the "weakest link" principle, selecting the minimum TF score among the top 3 snippets. This results in the methods BM25+TF-SP and MonoT5+TF-SP. We denote by +CE that the 3 snippets are further re-ranked by a cross-encoder. In addition, we compare the performance of these systems to the cross-encoder's performance when only evaluating the first snippet of each document (which resembles the naïve application of a cross-encoder). These results are denoted by BM25+CE and MonoT5+CE.

To measure the performance of the approaches, we utilize normalized discounted cumulative gain at 10 (NDCG@10) and mean reciprocal rank (MRR). We conduct our tests on the ClueWeb12 [8] and ClueWeb09 [9] datasets, which differ in document size: ClueWeb12 has an average document size of 5641.7 tokens, and ClueWeb09 has an average document size of 1132.6 tokens.



(a) ClueWeb09 (2011)

(b) ClueWeb12 (2013)

Figure 2: Experimental results on different datasets, blue bars denote NDCG@10, while red bars indicate MRR.

The results for the two datasets are plotted in Fig. 2. Our approach of cross-encoder re-ranking with TF-pre-ranked snippets achieves the best performance in both metrics across all our tested datasets (see Appendix B.2 for diagrams of other evaluated datasets). The impact of our TF-ranked snippet pre-selection is relatively high on ClueWeb12 with long documents, while it is more marginal on ClueWeb09. This highlights the importance of snippet pre-selection for longer documents. ClueWeb09 consists of approximately 6 snippets, and ClueWeb12 consists of approximately 23 snippets per document. We assume for our naïve snippet generation approach that information is equally spread throughout a document. A cross-encoder taking the first snippet as input is assumed to capture more relevant information of a document with a size that is closer to the cross-encoder input size. This also explains why MonoT5 scores better on the shorter dataset, especially in comparison to BM25, since MonoT5 also suffers from a limited input size. This proves that there is a need to address the problem of limited input size, especially in large documents like those in ClueWeb12. That information is not always equally spread over a document, like we assumed for our snippet generation, can be concluded when comparing Figs. 4b and 4c. This raises the need of a more advanced approach for snippet generation.

3.3. Summary

Overall, our results show that selecting top-k pre-ranked snippets is a viable approach to tackle the problem of input size restrictions on Transformer-based retrieval systems. Especially, cross-encoders can benefit from this approach since they are inefficient on large documents. Further testing to edge out efficiency and reduce context loss with snippets will be required. Also, it would be beneficial to test multiple pre-ranking systems and values of k for top-k snippet selection. The code for this part can be found in the accompanying repository[2].

4. Query Expansion and User Query Variants using Large Language Models

Query Expansion and User Query Variants are two common methods to increase the recall of an IR system [10, 11, 12]. Both methods are based on modifying the query to include more related keywords, thereby causing the IR system to score relevant documents higher. In addition to conventional techniques such as the Kullback-Leibler Divergence (KL) [13, 14] or Relevance Model 3 (RM3) [15], recent approaches have embraced the utilization of Large Language Models (LLMs). In this part, we employ various prompts to generate improved and expanded queries using LLMs [16, 17], in particular, GPT-3.5[3], Llama2 [18] and FLAN-UL2 [19].

4.1. Methodology

LLMs have previously been in use for the task of query expansion and studies have been conducted using various methods and language models [16, 20, 21, 22]. Wang et al. [21] employ *query2doc*, a method where the LLM generates a document for a given query, which is then used as Pseudo-Relevance Feedback (PRF). Jagerman et al. [16] follow a similar approach but extend the experiments to include alternative LLMs and additional prompt types. All

[2]https://github.com/tira-io/workshop-on-open-web-search-tu-dresden-02
[3]https://platform.openai.com/docs/models/gpt-3-5-turbo

Model	Temperature	min. Tokens	max. Tokens	Quantization	Parameters
Llama 2	1.1	10	200	4 bit	7B
FLAN-UL2	0.5	10	200	8 bit	20B
GPT-3.5 Turbo	0.5	–	200	–	175B

Table 1
Model parameters, for GPT-3.5 the parameter min. tokens is unavailable in the API

previous work demonstrates improvements across different datasets. In order to weigh the original query more heavily, multiple concatenations of the original query q with a single instance of the LLM's output may be used [16, 21]. The resulting expanded query is of the form $q' = concat(\{q\} * n, LLM_{out})$, where n is the number of times q is concatenated with itself, and LLM_{out} is the LLM-generated version of q. We adopt this approach in our work with $n = 5$ and employ modified versions of the prompt types suggested by Jagerman et al. [16]: Chain of Thoughts (CoT) where the model is prompted to document its thought process, Query to Expansion with Zero-Shot prompting (Q2E/ZS) where the model should reformulate the query directly, and Query to Expansion with Few-Shot prompting (Q2E/FS), where three examples for the desired query format are provided to the model. For the exact prompt format used, see Appendix C.3. It should be noted that the prompt for Q2E/ZS differs between the models. While GPT-3.5 and FLAN were prompted to generate five similar queries, Llama was asked to answer the query. Apart from this difference, the prompts for in all experiments are similar and comparable.

Initially, the query, along with the prompt, is fed into the LLM, and its response is concatenated with the original query ($n = 5$). For evaluation, the Recall@1000 metric of the original and modified queries is compared on the given dataset using BM25. The specific LLMs in use are GPT-3.5[45], Llama 2 [18] and FLAN-UL2 [19]. Llama 2 and FLAN-UL2 were run locally. Table 1 shows the model configurations that we used in our experiments. The temperature values were chosen empirically in a way such that model outputs are roughly similar. The lower and upper token limitations prevent generation edge cases such as empty responses or endless output, while still allowing for expressive responses. Local models had to fit GPU memory constraints. Hence, we had to employ the quantized versions of the models. We conducted experiments for the prompt types presented above: *CoT*, *Q2E/FS*, and *Q2E/ZS*. While FLAN-UL2 and GPT-3.5 can be prompted without further changes, Llama 2 requires the chat-prompt to follow a pre-defined format, our version of which can be found in the project's repository[6]. We utilize BM25 as the retrieval system in the default configuration of the Tira-framework [23]. The query expansion baselines consist of an unmodified BM25, BM25 with Kullback-Leibler Divergence (KL) and BM25 with RM3.

	Baseline			CoT			Q2E/FS			Q2E/ZS		
	BM25	KL	RM3	FLAN	Llama	GPT	FLAN	Llama	GPT	FLAN	Llama	GPT
Avg	0.66	0.67	**0.69**	0.68	0.67	**0.69**	0.66	0.67	0.67	0.67	0.67	0.68
Avg_{easy}	0.72	0.73	0.75	0.76	0.76	**0.78**	0.73	0.74	0.73	0.74	0.76	0.76

Table 2
Recall@1000 evaluation results. The best value across different configurations is **bolded**. Grey values failed to outperform the best baseline effectiveness. Avg denotes the arithmetic mean scores all 18 datasets. Avg_{easy} excludes the cord19, longeval and medline datasets.

4.2. Evaluation

We measure the recall, which is aggregated over 18 datasets per model, and per prompt type. The datasets cover a range of diverse topics and were provided as part of TIRA [24] / TIREx [25]. The aggregated results can be observed in Table 4.2. Avg_{easy} excludes evidently (cord19, longeval) and presumably (medline) difficult datasets. This highlights the difficulties LLMs experience on specific datasets, especially domain-specific ones: Excluding those, CoT+OpenAI GPT-3.5 Turbo (GPT) now performs 0.03 points better than baseline models. Note that the two *Avg* rows cannot be compared to one another, as baseline scores have also shifted due to the exclusion of generally low-performing datasets. Detailed results for each dataset can be found in Appendix C.1. For our query expansion approaches, it is evident that the choice of prompt has a large impact on recall performance. The combination of CoT and GPT consistently yields the highest recall in absolute numbers. However, with other prompt types such as Q2E/ZS and Q2E/FS, GPT also frequently achieves the highest recall per dataset, albeit less frequently compared to CoT. In this regard, our results are consistent with those reported in [16]. Although CoT generally performs the best, it exhibits poorer results than the baselines in datasets such as cord19 or the longeval datasets. In these cases, Q2E/ZS and Q2E/FS emerge as better choices, but are still commonly outperformed by the baseline models.

Q2E/FS exhibits less convincing effectiveness, presumably because it mimics the relatively short responses of example queries through the Q2E/FS method, resulting in short queries with few new keywords. Q2E/ZS behaves similarly. Although the responses of the LLMs are longer compared to Q2E/FS, as the LLMs do not conform to the rather short examples, the generated responses are overall less extensive than those of CoT, likely resulting in inferior effectiveness. Considering the longeval datasets and cord19, it is evident that they contain either very general or highly specific queries. In the case of nonspecific queries, there is a risk that they may be muddled by the consequently more general, and in the case of CoT, extensive responses from the LLMs. This effect might potentially be reversed by conveying the user intent to the LLM, indicating whether, for instance, in the case of the query "car," the user intends to buy one or have it repaired. With domain-specific queries, it is plausible that models were trained with insufficient knowledge on the subject, resulting in subpar effectiveness.

While our main evaluation is conducted using recall@1000, we also evaluated nDCG@10.

[4]https://platform.openai.com/docs/models/gpt-3-5-turbo
[5]https://platform.openai.com/docs/api-reference/
[6]https://github.com/tira-io/workshop-on-open-web-search-tu-dresden-03

Name	Formula
Flesch Reading Ease [29]	$206.835 - 1.015 * \left(\frac{\text{Word Count}}{\text{Sentence Count}}\right) - 84.6 * \left(\frac{\text{Syllable Count}}{\text{Word Count}}\right)$
Flesch Kincaid grade level [29]	$0.39 * \left(\frac{\text{Word Count}}{\text{Sentence Count}}\right) + 11.8 * \left(\frac{\text{Syllable Count}}{\text{Word Count}}\right) - 15.59$
Gunning FOG [29, 30]	$0.4 * \left[\left(\frac{\text{Word Count}}{\text{Sentence Count}}\right) + 100\left(\frac{\text{Complex Word Count}}{\text{Word Count}}\right)\right]$
SMOG Index [31]	$1.0430 * \sqrt{\text{Complex Word Count} * \frac{30}{\text{Sentence Count}}} + 3.1291$
Automated Readability Index [29, 32]	$4.71 * \left(\frac{\text{Character Count}}{\text{Word Count}}\right) + 0.5\left(\frac{\text{Word Count}}{\text{Sentence Count}}\right) - 21.43$
Coleman-Liau Index [33]	$5.88 * \left(\frac{\text{Character Count}}{\text{Word Count}}\right) - 29.6 * \left(\frac{\text{Sentence Count}}{\text{Word Count}}\right) - 15.8$
LIX [34]	$\frac{\text{Word Count}}{\text{Sentence Count}} + 100 * \left(\frac{\text{Long Word Count}}{\text{Word Count}}\right)$
RIX [34]	$\frac{\text{Long Word Count}}{\text{Sentence Count}}$

Table 3
Implemented Text Features with the respective formulas. Syllable count and word count were implemented using the provided tools by the Text Feature Libraries Textstat and spaCy.

The results for this metric are detailed in Table 4 in the appendix. Overall, our conclusions for nDCG are similar to those for recall. The generations for each model and each prompt are publicly available in our repository[7].

4.3. Summary

In this part, we generated different versions of query expansions using three LLMs and three prompt templates. We were able to demonstrate that LLMs are capable of improving the recall of user queries. The combination of the prompt CoT alongside GPT proves to be the most promising, improving recall scores by up to 15%. Future research could focus on further templates for using the generated expansions since we only evaluated the $qqqqq, response$-format.

5. Text Features

Text Features are quantified metrics describing syntactic or semantic features of natural language. An example is the readability of a text, useful for returning user-dependent search results. A search engine targeted to school children should return results with a high readability score, whereas a search engine with domain experts as target audience will also include texts with low readability scores. Additionally, this could be used to filter out noisy websites.

This Open Web Search component[8] incorporates two tools for computing text features, namely Textstat [26] and textdescriptives [27] from spaCy [28]. SpaCy's Text Feature analysis is more comprehensive than the one in Textstat, but is less efficient. Per design of the pipeline approach of SpaCy many things are computed in the background, from which only a few are required for the calculation of the text features. This overhead results in a longer runtime which should be considered. Table 3 displays the implemented text features.

[7]https://github.com/tira-io/workshop-on-open-web-search-tu-dresden-03/tree/main/src/generated
[8]https://github.com/tira-io/workshop-on-open-web-search-tu-dresden-04

Additional contributions besides the integration of the text features components include examining a potential correlation between text features and documents evaluated as relevant by ranked retrieval models. For easier exploration of the document corpus we provide an interactive Jupyter Notebook showing correlation graphs between Ranked Retrieval and Text Features, applicable to arbitrary datasets, as well as the analysis of correlations between Ranked Retrieval and Text Measures.

5.1. Evaluation

To verify the capability to differentiate between levels of Readability, unit tests were used. The test data consists of multi-sentence snippets from web pages. These were categorized by difficulty in the following categories (including the amount of test documents): children (3), teenagers (3), academic (3), and simple language (2), depending on what demographic the source was directed to. Initial tests involved a project member assessing the reading level of excerpts and comparing their assessments to the classifications provided by the automated measures, thus proving correct usage of the used text feature libraries at least for the readability scores.

Compared to human assessments, the automated Text Measures often overestimated the reading level, possibly failing to capture the complexities of human reading abilities within their respective indexes. Large-scale dataset computations further highlighted the discrepancies between predicted and human-classified reading levels, corroborating these findings. Despite the observed differences in assessment, the data suggested an inverse proportional relationship between comprehension levels and readability measures.

5.2. Experiment Design

The experiments were run on the "antique/test" [35] dataset from the ir_datasets collection [36]. Based on TIRA [37] ranked retrieval models were used to create top-10 results.

5.2.1. Correlations between ranked retrieval and text feature readability

A primary objective of our project was to investigate whether the ranking of relevant documents by ranked retrieval models correlates to document Readability.

5.2.2. Readability of Top 10

First we looked at the top 10 retrieved documents for all queries across multiple retrieval models, the resulting distributions are displayed in Figure 3. The majority of results, assessed using the Flesch Reading Ease, indicates comprehension levels at or below an eighth-grade level, implying a high degree of readability. The high degree of readability was consistently observed across multiple retrieval models. Compared to the overall readability across all documents in a collection, we found that some retrieval models like SBERT or MonoT5 indeed result in a higher readability in the retrieved documents compared to the rest of the corpora, suggesting a potential relationship between relevancy and readability, whereas other retrieval models such as BM25 do not share this characteristic.

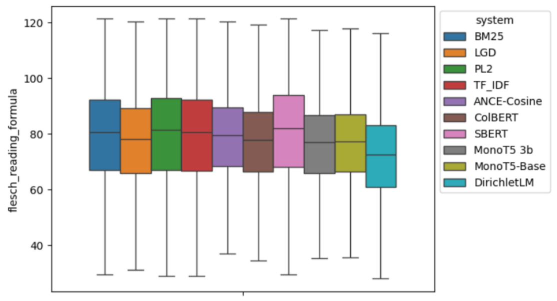


Figure 3: Box plot of Flesch Reading Ease for Top 10 documents of all queries over selected ranked retrieval models

The results of our analysis revealed an inverse correlation between readability and necessary reading comprehension levels. We found that relevant documents generally exhibit higher readability, except for complex queries, which tend to retrieve domain-specific documents with lower readability. Another notable finding was that the Top 10 retrieved results from retrieval models demonstrated high readability. The correlation between readability and corpus readability was found to be corpus-dependent, and retrieval models often retrieved highly readable documents within the Top 10 results.

5.3. Possibilities for Future Work

Future research endeavors could involve utilizing the extracted text features to re-rank documents retrieved by Retrieval Models, thereby enhancing the relevance ranking by incorporating Readability Metrics.

Additionally, optimizing the project could involve matching the readability level of the retrieved documents to the readability level of the user based on the readability score of the query itself.

Furthermore, an unexplored extension could examine whether partitioning the dataset based on predefined readability score thresholds facilitates the generation of demographic-specific results.

6. Conclusion and Future Work

In this work, we have introduced four diverse components that can be integrated into a larger retrieval pipeline: Genre Classification, Text Snippet Extraction, Query Expansion, and Text Feature Extraction. We have provided short use cases for each component as part of our evaluations that demonstrate the benefits of using the developed components. Even though our main focus was implementing proof-of-concepts of our components, we are sure that future research can easily be built upon our efforts. We hope that our components can be integrated seamlessly into larger IR research projects and facilitate the usage of our built-in methods, and are looking forward to the future of the open web search ecosystem.

7. Acknowledgments

We would like to express our gratitude to the Open Search Foundation for organizing the WOWS 2024 and especially Maik Fröbe, who supported us and our student teams in organizing and conducting our Hackathon which made this submission possible.

In addition, the authors gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Education and Research and the state governments participating in the NHR (www.nhr-verein.de/unsere-partner).

References

[1] S. Farzana, M. Fröbe, M. Granitzer, G. Hendriksen, D. Hiemstra, M. Potthast, S. Zerhoudi, 1st International Workshop on Open Web Search (WOWS), in: Advances in Information Retrieval. 46th European Conference on IR Research (ECIR 2024), Lecture Notes in Computer Science, Springer, 2024.

[2] S. Meyer zu Eissen, B. Stein, Genre classification of web pages, in: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 256–269.

[3] A. Z. Broder, A taxonomy of web search, SIGIR Forum 36 (2002) 3–10. URL: https://doi.org/10.1145/792550.792552. doi:10.1145/792550.792552.

[4] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, C. Ré, Snorkel: Rapid training data creation with weak supervision, in: Proceedings of the VLDB endowment. International conference on very large data bases, volume 11, NIH Public Access, 2017, p. 269.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[6] R. Nogueira, K. Cho, Passage Re-ranking with BERT, 2020. URL: http://arxiv.org/abs/1901.04085. doi:10.48550/arXiv.1901.04085.

[7] J. Dalton, C. Xiong, J. Callan, TREC CAsT 2019: The Conversational Assistance Track Overview, 2020. URL: http://arxiv.org/abs/2003.13624. doi:10.48550/arXiv.2003.13624.

[8] J. Callan, The lemur project and its clueweb12 dataset, in: Invited talk at the SIGIR 2012 Workshop on Open-Source Information Retrieval, 2012.

[9] J. Callan, M. Hoy, C. Yoo, L. Zhao, Clueweb09 data set, 2009.

[10] J. J. R. Jr., Relevance feedback in information retrieval. The SMART retrieval system: experiments in automatic document processing (1971).

[11] R. R. Korfhage, To see, or not to see—is that the query?, in: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval, 1991, pp. 134–141.

[12] J. Yang, R. R. Korfhage, E. Rasmussen, Query improvement in information retrieval using

genetic algorithms–a report on the experiments of the trec project, in: Proceedings of the Text REtrieval Conference (TREC-1), 1993, pp. 31–58.

[13] S. Kullback, R. A. Leibler, On Information and Sufficiency, The Annals of Mathematical Statistics 22 (1951) 79 – 86. URL: https://doi.org/10.1214/aoms/1177729694. doi:10.1214/aoms/1177729694.

[14] F. Raiber, O. Kurland, Kullback-leibler divergence revisited, in: Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 117–124. URL: https://doi.org/10.1145/3121050.3121062. doi:10.1145/3121050.3121062.

[15] V. Lavrenko, W. B. Croft, Relevance-based language models, in: ACM SIGIR Forum, volume 51, ACM New York, NY, USA, 2017, pp. 260–267.

[16] R. Jagerman, H. Zhuang, Z. Qin, X. Wang, M. Bendersky, Query expansion by prompting large language models, 2023. arXiv:2305.03653.

[17] M. Alaofi, L. Gallagher, M. Sanderson, F. Scholer, P. Thomas, Can generative llms create query variants for test collections? an exploratory study, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 1869–1873. URL: https://doi.org/10.1145/3539618.3591960. doi:10.1145/3539618.3591960.

[18] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open foundation and fine-tuned chat models, 2023. arXiv:2307.09288.

[19] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, S. Zheng, et al., Ul2: Unifying language learning paradigms, in: The Eleventh International Conference on Learning Representations, 2022.

[20] V. Claveau, Neural text generation for query expansion in information retrieval, in: WI-IAT 2021 - 20th IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Proceedings of the WI-IAT Conference, IEEE, Melbourne, Australia, 2021, pp. 1–8. URL: https://hal.science/hal-03494692. doi:10.1145/3486622.3493957.

[21] L. Wang, N. Yang, F. Wei, Query2doc: Query expansion with large language models, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 9414–9423.

[22] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, J.-R. Wen, Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 2825–2835.

[23] M. Fröbe, J. H. Reimer, S. MacAvaney, N. Deckers, S. Reich, J. Bevendorff, B. Stein, M. Hagen, M. Potthast, The information retrieval experiment platform, in: Proceedings of the

46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, ACM, 2023. URL: http://dx.doi.org/10.1145/3539618.3591888. doi:10.1145/3539618.3591888.

[24] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, M. Potthast, Continuous Integration for Reproducible Shared Tasks with TIRA.io, in: J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2023, pp. 236–241. URL: https://link.springer.com/chapter/10.1007/978-3-031-28241-6_20. doi:10.1007/978-3-031-28241-6_20.

[25] M. Fröbe, J. H. Reimer, S. MacAvaney, N. Deckers, S. Reich, J. Bevendorff, B. Stein, M. Hagen, M. Potthast, The Information Retrieval Experiment Platform, in: H.-H. Chen, W.-J. E. Duh, H.-H. Huang, M. P. Kato, J. Mothe, B. Poblete (Eds.), 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023), ACM, 2023, pp. 2826–2836. URL: https://dl.acm.org/doi/10.1145/3539618.3591888. doi:10.1145/3539618.3591888.

[26] S. Bansal, textstat, https://github.com/textstat/, 2016.

[27] L. Hansen, L. R. Olsen, K. Enevoldsen, TextDescriptives: A Python package for calculating a large variety of metrics from text, Journal of Open Source Software 8 (2023) 5153. URL: https://joss.theoj.org/papers/10.21105/joss.05153. doi:10.21105/joss.05153.

[28] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python (2020). doi:10.5281/zenodo.1212303.

[29] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, B. S. Chissom, Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel (1975).

[30] J. Bogert, In defense of the fog index, The Bulletin of the Association for Business Communication 48 (1985) 9–12.

[31] G. H. Mc Laughlin, Smog grading-a new readability formula, Journal of reading 12 (1969) 639–646.

[32] R. Senter, E. A. Smith, Automated readability index, Technical Report, Technical report, DTIC document, 1967.

[33] M. Coleman, T. L. Liau, A computer readability formula designed for machine scoring., Journal of Applied Psychology 60 (1975) 283.

[34] J. Anderson, Lix and rix: Variations on a little-known readability index, Journal of Reading 26 (1983) 490–496. URL: http://www.jstor.org/stable/40031755.

[35] H. Hashemi, M. Aliannejadi, H. Zamani, B. Croft, Antique: A non-factoid question answering benchmark, in: ECIR, 2020.

[36] S. MacAvaney, A. Yates, S. Feldman, D. Downey, A. Cohan, N. Goharian, Simplified data wrangling with ir_datasets, in: F. Diaz, C. Shah, T. Suel, P. Castells, R. Jones, T. Sakai (Eds.), SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, ACM, 2021, pp. 2429–2436. URL: https://doi.org/10.1145/3404835.3463254. doi:10.1145/3404835.3463254.

[37] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, M. Potthast, Continuous Integration for Reproducible Shared Tasks with TIRA.io, in:

J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2023, pp. 236–241. URL: https://link.springer.com/chapter/10.1007/978-3-031-28241-6_20. doi:10.1007/978-3-031-28241-6_20.

[38] I. Montani, M. Honnibal, A. Boyd, S. V. Landeghem, H. Peters, spaCy: Industrial-strength Natural Language Processing in Python, 2023. URL: https://zenodo.org/records/10009823. doi:10.5281/zenodo.10009823.

[39] C. Macdonald, N. Tonellotto, Declarative experimentation in information retrieval using pyterrier, in: Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval, 2020, pp. 161–168.

[40] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: State-of-the-Art Natural Language Processing, in: Q. Liu, D. Schlangen (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. URL: https://aclanthology.org/2020.emnlp-demos.6. doi:10.18653/v1/2020.emnlp-demos.6.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

A. Hackathon

The paper's work was carried out by students from TU Dresden as part of a one-week hackathon. The workshop was open to students in the Computer Science program and related fields, and they could earn ECTS credit points for lab work. The hackathon was advertised on the mailing lists of the beginner Information Retrieval courses from the past three years. Interested students could fill a survey indicating their preferred timeframe for the hackathon.

After a date was decided, 10 students signed up for the hackathon, three from the Bachelor's program and seven from the Master's program. The university supervisors prepared four topics, which were advertised beforehand, and the students signed up for their preferred topics. The text features topic was designated to the 3 Bachelor students. The Master students were provided with a peer-reviewed research paper as additional material, which they were required to read and understand before the hackathon.

On the first day of the hackathon, an invited member of the Open Web Search project provided a brief introduction to the Open Web Search ecosystem and TIRA/TIREx. Following this, the teams worked on their components, with supervisors providing guidance through daily check-ins. On the fifth and final day of the hackathon, a short presentation from each team was held. Following the hackathon, the students were requested to prepare a report on their work, which served as the basis for this paper.

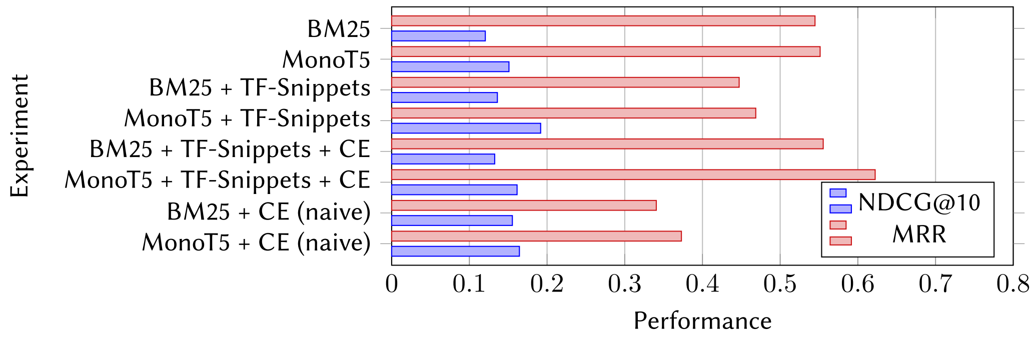
In retrospective, the short amount of time, one week, motivated the students to work diligently on their project. However, at the end of the week, the students had several open ideas for future work which they could not finish in time. Therefore, more time, even a few days more, might be beneficial for the next iteration of the hackathon. The size of the group ranged between two and three members. The small group size facilitated the organization within each group and kept the management overhead small. The topics of the hackathon were aligned with the basics gained during the Information Retrieval course, but required also reading additional literature and research.

B. Text Snippet Extraction

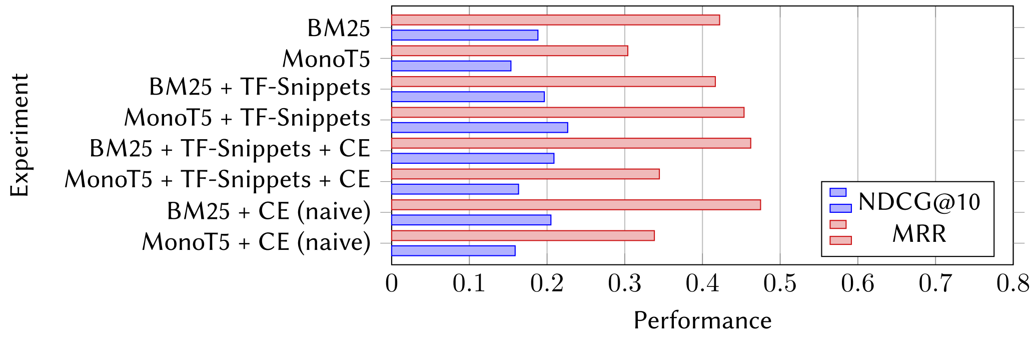
B.1. Implementation

To implement the described re-ranking steps, we utilized several Python libraries, detailed below to facilitate reproducibility. For snippet extraction in Step 1, we adapted the *Spacy-PassageChunker* class from the *corpus_processing* package, as provided by Dalton et al. [7], to allow for variable snippet sizes. The class requires *spaCy* [38]; we used version 3.3.0 for our implementation. The snippet pre-ranking in Step 2 was implemented using *PyTerrier* [39], version 0.10.0. For Step 4 we utilized *ms-marco-MiniLM-L-6-v2* which has been published on *HuggingFace.co* [40]. To embed the model into our project, we used the *transformers* library [40], version 4.38.2, and the *PyTorch* library [41], version 2.2.0. The results of the preparation steps are accessible via TIRA [24] / TIREx [25].

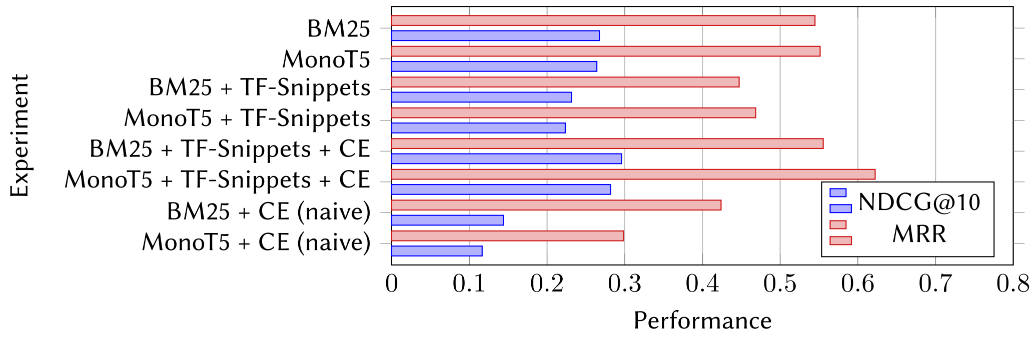
B.2. Results on other evaluated datasets



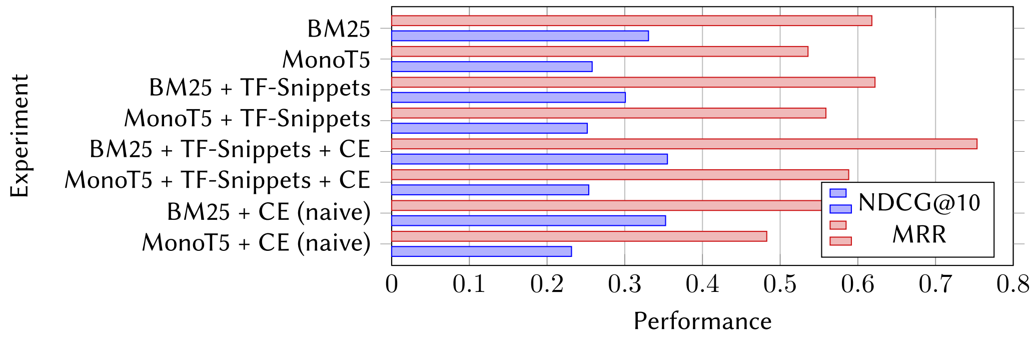
(a) Clueweb09 (2009)



(b) Clueweb09 (2010)



(c) Clueweb09 (2012)



(d) Clueweb12 (2014)

Figure 4: Experimental results on other datasets

C. Query Expansion

C.1. Detailed Results for Recall

Dataset	Baseline			CoT			Q2E/FS			Q2E/ZS		
	BM25	KL	RM3	FLAN	Llama	GPT	FLAN	Llama	GPT	FLAN	Llama	GPT
antique	0.79	0.78	0.79	0.80	0.79	**0.81**	0.79	0.80	0.80	0.80	0.80	0.79
argsme-2020-1	0.88	0.88	**0.89**	0.88	0.87	0.88	0.88	0.88	0.88	0.87	0.88	0.88
argsme-2021-1	0.94	0.95	0.95	0.95	0.95	**0.96**	0.94	0.94	0.94	**0.96**	0.95	**0.96**
cord19	0.43	0.41	**0.44**	0.38	0.31	0.33	0.41	0.42	0.43	0.41	0.36	0.42
cranfield	0.61	0.64	0.69	**0.70**	0.69	**0.70**	0.62	0.65	0.62	0.61	0.69	**0.70**
longeval-heldout	0.63	0.65	0.66	0.64	0.62	0.62	0.62	0.65	**0.67**	0.65	0.52	0.64
longeval-long-september	0.71	0.71	**0.72**	0.69	0.63	0.66	0.71	0.71	0.71	0.69	0.68	0.70
longeval-short-july	0.68	0.69	**0.70**	0.67	0.63	0.65	0.68	0.69	0.69	0.67	0.66	0.68
longeval-train	0.71	0.72	0.72	0.70	0.64	0.67	0.70	0.72	**0.73**	0.70	0.67	0.71
medline-2004	0.45	0.49	**0.50**	0.45	0.46	0.47	0.45	0.46	0.45	0.46	0.46	0.43
medline-2005	0.50	0.51	0.51	0.51	0.51	**0.53**	0.50	0.51	0.51	0.50	0.52	**0.53**
medline-2017	0.59	0.60	0.62	0.62	0.65	**0.71**	0.59	0.62	0.59	0.59	0.63	0.64
medline-2018	0.72	0.78	0.79	0.74	0.75	**0.81**	0.71	0.73	0.73	0.73	0.75	0.75
msmarco-pasage-2019	0.74	0.75	0.75	0.79	0.82	**0.86**	0.75	0.76	0.74	0.79	0.81	0.78
msmarco-passage-2020	0.75	0.80	0.79	0.81	0.81	**0.85**	0.76	0.76	0.76	0.80	0.78	0.77
nfcorpus-test	0.32	0.46	0.44	0.44	0.45	**0.48**	0.35	0.38	0.37	0.37	0.46	0.45
tip-of-the-tongue-dev	0.52	0.38	0.52	0.51	0.51	**0.57**	0.52	0.53	0.54	0.52	0.55	0.54
vaswani	0.93	0.94	0.94	0.94	0.95	**0.96**	0.94	0.94	0.93	0.94	0.95	0.94
Avg	0.66	0.67	**0.69**	0.68	0.67	**0.69**	0.66	0.67	0.67	0.67	0.67	0.68
Avg$_{easy}$	0.72	0.73	0.75	0.76	0.76	**0.78**	0.73	0.74	0.73	0.74	0.76	0.76

Table 4
Recall@1000 evaluation results. For each dataset, the best value across different configurations is **bolded**. Grey values failed to outperform the best baseline performance. The arithmetic mean scores can be found at the bottom. Avg$_{easy}$ excludes the cord19, longeval and medline datasets.

C.2. Detailed Results for nDCG

Dataset	Baseline			CoT			Q2E/FS			Q2E/ZS		
	BM25	KL	RM3	FLAN	Llama	GPT	FLAN	Llama	GPT	FLAN	Llama	GPT
antique	0.51	0.49	0.5	0.5	0.5	0.51	0.51	0.52	0.52	0.5	**0.53**	0.5
argsme-2020-1	0.3	0.32	0.32	0.35	0.4	**0.42**	0.31	0.33	0.31	0.3	0.39	0.37
argsme-2021-1	0.51	0.52	0.54	0.55	0.56	0.56	0.51	0.51	0.52	0.53	**0.57**	0.56
cord19	0.59	0.52	**0.61**	0.38	0.42	0.5	0.49	0.58	0.57	0.53	0.48	0.5
cranfield	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**
longeval-heldout	0.16	0.16	**0.18**	0.16	0.14	0.16	0.17	0.16	0.17	0.15	0.09	0.16
longeval-long-september	0.18	0.18	**0.19**	0.17	0.14	0.16	0.18	0.18	**0.19**	0.17	0.16	0.17
longeval-short-july	**0.18**	0.17	**0.18**	0.16	0.14	0.15	**0.18**	**0.18**	**0.18**	0.16	0.16	0.17
longeval-train	**0.18**	0.17	0.17	0.15	0.13	0.15	0.17	0.17	**0.18**	0.15	0.14	0.17
medline-2004	0.34	**0.35**	**0.35**	**0.35**	0.32	0.34	**0.35**	**0.35**	0.34	0.34	**0.35**	0.34
medline-2005	0.38	0.38	0.38	0.38	0.42	0.41	0.39	0.39	0.39	0.39	0.4	**0.43**
medline-2017	0.28	0.31	0.3	0.3	0.34	**0.36**	0.28	0.3	0.28	0.28	0.3	0.34
medline-2018	0.43	0.46	0.48	0.45	0.44	**0.53**	0.4	0.43	0.42	0.41	0.44	0.46
msmarco-passage-2019	0.48	0.52	0.51	0.58	0.57	**0.63**	0.5	0.5	0.48	0.55	0.6	0.52
msmarco-passage-2020	0.49	0.5	0.49	0.58	0.57	**0.59**	0.49	0.5	0.5	0.57	0.53	0.51
nfcorpus-test	0.27	0.27	0.28	0.28	**0.29**	**0.29**	0.27	0.28	0.27	0.27	0.28	0.28
tip-of-the-tongue-dev	0.1	0.06	0.09	0.1	0.11	**0.15**	0.11	0.11	0.1	0.11	0.11	0.11
vaswani	0.45	0.44	0.45	0.45	0.43	**0.46**	0.44	0.45	**0.46**	0.45	0.45	**0.46**
Avg	0.32	0.32	0.33	0.33	0.33	**0.35**	0.32	0.33	0.33	0.33	0.33	0.34
Avg$_{easy}$	0.35	0.35	0.35	0.38	0.38	**0.4**	0.35	0.36	0.35	0.37	0.39	0.37

Table 5
NDCG@10 evaluation results. **Bold** indicates the strongest score. Grey results fell below the best baseline. Avg$_{easy}$ excludes the cord19, longeval and medline datasets.

C.3. Prompts

Method	Prompt

CoT	`f'Answer the following query:'`
	`f''`
	`f'{q}'`
	`f''`
	`f'Give the rationale before answering.'`

Q2E/FS	`f'For every query, suggest a similar query:'`
	`f''`
	`f'Original query: How to tie a windsor knot?'`
	`f'Similar query: Instructions for tying a windsor knot'`
	`f''`
	`f'Original query: How is the weather tomorrow morning?'`
	`f'Similar query: Weather tomorrow morning'`
	`f''`
	`f'Original query: Simple vegan cooking recipes'`
	`f'Similar query: What are some delicious and simple vegan cooking recipes?'`
	`f''`
	`f'Original Query: {q}'`
	`f'Similar Query:'`

Q2E/ZS	`f'Suggest 5 queries that are similar to the following query:'`
	`f''`
	`f'Query: {q}'`

Table 6
Prompt formats used for GPT and Google Flan-UL2 (Flan).

Method	Prompt

CoT	`f'<s>[INST] <<SYS>>\n'`
	`f'Be short and concise, 100 words max. Answer in full sentences, while briefly writing down your steps towards the response.'`
	`f'\n<</SYS>>\n\n'`
	`f'{q}'`
	`f' [/INST]'`

Q2E/FS	`f'<s>[INST] <<SYS>>\n'`
	`f'Be short and concise, 100 words max. Answer in full sentences, while briefly writing down your steps towards the response.'`
	`f'\n<</SYS>>\n\n'`
	`f'{f'For every query, suggest a similar query:'`
	`f''`
	`f'Original query: How to tie a windsor knot? [/INST]'`
	`f'Similar query: Instructions for tying a windsor knot </s>'`
	`f''`
	`f'<s>[INST] Original query: How is the weather tomorrow morning? [/INST]'`
	`f'Similar query: Weather tomorrow morning </s>'`
	`f''`
	`f'<s>[INST] Original query: Simple vegan cooking recipes [/INST]'`
	`f'Similar query: What are some delicious and simple vegan cooking recipes? </s>'`
	`f''`
	`f'<s>[INST] Original query: {q} [/INST]'`
	`f'Similar query:'}'`
	`f' [/INST]'`

Q2E/ZS	`f'<s>[INST] <<SYS>>\n'`
	`f'Answer the following query. Be short and concise, 50 words at max. Answer in full sentences.'`
	`f'\n<</SYS>>\n\n'`
	`f'{q}'`
	`f' [/INST]'`

Table 7
Prompt formats used for Meta Llama 2 7B Chat (Llama). Note the necessity for a system prompt and the additional formatting sequences due to the instruction fine-tuning of Llama-Chat. Prompts were modified to fit Llama's behaviour.