

CIR at TREC 2024 RAG: Task 2 - Augmented Generation with Diversified Segments and Knowledge Adaption

Jüri Keller

TH Köln - University of Applied
Sciences

Cologne, Germany
jueri.keller@th-koeln.de

Björn Engemann

TH Köln - University of Applied
Sciences

Cologne, Germany
bjoern.engemann@th-koeln.de

Fabian Haak

TH Köln - University of Applied
Sciences

Cologne, Germany
fabian.haak@th-koeln.de

Philipp Schaer

TH Köln - University of Applied
Sciences

Cologne, Germany
philipp.schaer@th-koeln.de

Hermann Kroll

Institute for Information Systems,
TU Braunschweig

Braunschweig, Germany
krollh@acm.org


Christin Katharina Kreutz

TH Mittelhessen - University of
Applied Sciences

Gießen, Germany
Herder Institute
Marburg, Germany
ckreutz@acm.org

Abstract

This paper describes the CIR team’s participation in the TREC 2024 RAG track for task 2, augmented generation. With our approach, we intended to explore the effects of diversification of the segments that are considered in the generation as well as variations in the depths of users’ knowledge on a query topic. We describe a two-step approach that first reranks input segments such that they are as similar as possible to a query while also being as dissimilar as possible from higher ranked relevant segments. In the second step, these reranked segments are relayed to an LLM, which uses them to generate an answer to the query while referencing the segments that have contributed to specific parts of the answer. The LLM considers the varying background knowledge of potential users through our prompts.

 github.com/irgroup/TREC2024-RAG-CIR

Keywords

Retrieval Augmented Generation (RAG), Diversity, Maximal Marginal Relevance (MMR), Reranking

1 Introduction

Retrieval-Augmented Generation (RAG) provides direct answers to users’ information needs that are grounded in potentially relevant documents. RAG comprises multiple steps that are reflected in the TREC RAG sub-tasks. We work on the Augmented Generation Task (Task 2), where we are given a list of segments ranked by relevance to a query topic. From these segments, we generate an answer that contains references back to the individual segments.

Usually, when reranking retrieved information for queries, users are expected to assess results until their information need is satisfied. In RAG, we assume that users want answers, not single result segments. Therefore, we derive that multiple segments containing the same information do not contribute much to form an answer. We expect a diversification of segments used in the generation step of RAG to potentially better reflect multiple different viewpoints on a topic. This variety should produce a more nuanced answer than

only considering segments that are as similar as possible to a query topic. Additionally, we expect diversified result lists to enable the usage of smaller models for the generation step, as fewer segments need to be handled compared to purely similarity-ranked result lists. As a side effect, such diversified input might help to address the lost in the middle phenomenon [5].

In order to effectively support users in fulfilling their information needs, generated answers should fit the depth of knowledge of a user. The perfect answer from an information retrieval perspective detailing different viewpoints might be useless for a user who is overstrained by unfamiliar domain jargon. Therefore, an adaption of the generated answer to the depth of knowledge of a user, similar to the efforts found in text simplicity [2–4], could be worthwhile.

In our approach, we consider both the diversification of segments used for the generation as well as the adaption of produced answers for the depth of knowledge of potential users. We rerank segments using MMR [1] and either Jaccard or cosine similarity. With four different prompting options, we model the users’ differing depths of knowledge.

2 Pipeline

Our pipeline consists of two parts: an optional reranking step for the pre-ranked segments and a generation step for generating the RAG output from reranked or pre-ranked segments. Figure 1 gives an overview of our approach.

2.1 Step 1: Reranking

We experiment with two variants: not reranking and reranking. In cases where we do not rerank, we use the top j pre-ranked segments as input for the generation step.

In cases where we rerank, we rerank the top j segments from the pre-ranked input to diversify the given ranked result list. Here, we apply the Maximum Marginal Relevance [1] (MMR) criterion. The key intuition behind MMR is to minimize redundant information while maintaining query relevance when selecting documents. As the authors note, this approach is particularly suitable for choosing documents for summarization. Texts are ranked such that they are

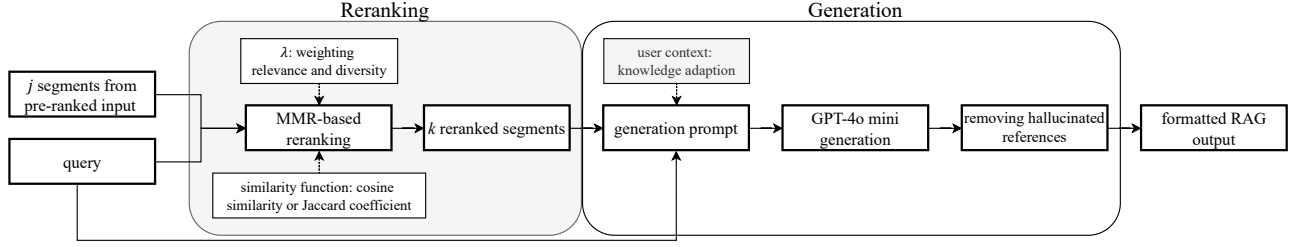


Figure 1: Overview of our approach for augmented generation with diversified segments and knowledge adaption. Gray-highlighted components are not used in all run configurations.

Variable	Explanation	Values
j	Number of considered segments from the pre-ranked segment list	100
$reranking$	Indication, if a reranking is conducted	Yes, No
λ	Weighting factor in MMR for trade-off between similarity of candidate to query and dissimilarity of candidate to ranked segments	0.25, 0.5, 0.75, 1
MMR_{sim}	Similarity calculation measure in MMR	Jaccard, Cosine
k	Number of reranked segments given to generation step	20, 50
$prompt$	Prompt used to generate the answer to an input query from the segments given to the generation step, the base prompt is extended with a user context, details in Table 2	P.0, P.1, P.2, P.3

Table 1: Variables and their explanations.

$prompt$	Prompt Text
Base	"Do retrieval augmented generation with the following texts\n" + USER_CONTEXT + "according to the following query: \n" + QUERY + "\n" + "make short and concise sentences while combining multiple texts into one sentence if they convey the same information\n" + "these are the texts, ordered by relevance with their ids:\n" + SEGMENTS + "\ngive the output as a json, where all sentences are separate and are assigned the ids of the texts, where the information appears\n" + ""answer': ['text': 'The frequency with which you should take your toddler to the potty depends on their readiness for potty training.', 'citations': [msmarco_passage_25_113697330, msmarco_passage_14_113668330], 'text': 'Some sources suggest that toddlers should be taken to the potty about three times a day: first thing in the morning, after mealtimes, and again before bedtime.', 'citations': [msmarco_passage_25_113697330, msmarco_passage_21_113697362]]""
P.0	""
P.1	"Please bear in mind that I am a beginner in this field and have little prior knowledge.\n"
P.2	"Please bear in mind that I am an expert in this field and have substantial prior knowledge.\n"
P.3	"Please bear in mind that I have some foundational knowledge in this field but am not deeply experienced.\n"

Table 2: Base prompt for the generation step consisting of the **general prompt**, **user context**, **query presentation and query**, **prompt details**, **text introduction and segments**, **the output method** and **a few-shot example**. Parts of the base prompt which are marked in bold are replaced: For **User_Context** either P.0, P.1, P.2 or P.3 can be inserted. For **QUERY** the query is inserted, for **SEGMENTS** the Python dictionary-formatted pairs of segment IDs and segment texts are inserted.

as similar as possible to the query and as dissimilar as possible from the segments that have already been ranked. The iterative structure of the result list, which maximizes the MMR criterion, is described by the following equation:

$$MMR = \arg \max_{D_i \in R \setminus S} \left[\lambda \cdot \text{Sim}_1(D_i, Q) - (1 - \lambda) \cdot \max_{D_j \in S} \text{Sim}_2(D_i, D_j) \right] \quad (1)$$

Here, D_i is a candidate item, R is the set of all candidates, S is the set of selected items, and Q is the query. $\text{Sim}_1(D_i, Q)$ measures

relevance between D_i and Q , while $\text{Sim}_2(D_i, D_j)$ measures diversity between D_i and selected items D_j . The parameter λ controls the balance, where λ emphasizes relevance and $1 - \lambda$ emphasizes diversity. A $\lambda=0.5$ weights both the similarity of a candidate segment to a query as well as the dissimilarity of a candidate segment to all already ranked segments equally. A $\lambda>0.5$ gives more weight to the similarity of a candidate segment to a query, while $\lambda<0.5$ gives more weight to the dissimilarity of a candidate segment to all already ranked segments.

As similarity function in our experiments we use cosine similarity or Jaccard coefficient for Sim_1 as well as Sim_2 .

#	Run ID	j	reranking	λ	MMR_sim	k	prompt	Priority
1	cir_gpt-4o-mini_Jaccard_50_0.5_100_301_p0	100	Yes	0.5	Jaccard	50	P.0	5
2	cir_gpt-4o-mini_Jaccard_50_1.0_100_301_p0	100	Yes	1.0	Jaccard	50	P.0	7
3	cir_gpt-4o-mini_Cosine_50_0.5_100_301_p1	100	Yes	0.5	Cosine	50	P.1	1
4	cir_gpt-4o-mini_Cosine_50_0.25_100_301_p1	100	Yes	0.25	Cosine	50	P.1	8
5	cir_gpt-4o-mini_Cosine_50_0.75_100_301_p1	100	Yes	0.75	Cosine	50	P.1	9
6	cir_gpt-4o-mini_Cosine_50_1.0_100_301_p1	100	Yes	1.0	Cosine	50	P.1	6
7	cir_gpt-4o-mini_Cosine_20_0.5_100_301_p1	100	Yes	0.5	Cosine	20	P.1	10
8	cir_gpt-4o-mini_Cosine_50_0.5_100_301_p2	100	Yes	0.5	Cosine	50	P.2	3
9	cir_gpt-4o-mini_Cosine_50_0.5_100_301_p3	100	Yes	0.5	Cosine	50	P.3	4
10	cir_gpt-4o-mini_no_reranking_50_0.5_100_301_p1	100	No	-	-	50	P.1	2

Table 3: Submitted run configurations.

When applying Jaccard, we tokenize texts, remove stopwords according to the nltk corpus stopword list for English, and stem the remaining tokens with the Porter stemmer. For cosine similarity we use text embeddings produced by the OpenAI TEXT-EMBEDDING-3-SMALL model¹. The number of reranked segments to return is k .

2.2 Step 2: Generation

We generate a RAG output for the queries and segments using GPT-4o mini² and generation prompts constructed from a set of parts, some of which are varied for different run configurations (see Table 2). The query is combined with the top k reranked segments in configurations using reranking (described in subsection 2.1) or the top j pre-ranked segments in cases where we do not use reranking. Depending on the run configuration, the model is asked to generate the output for a beginner, an expert, a user with some knowledge, or no specific user. Few-shot examples are provided, and the model is prompted to produce a JSON output both in the prompt and in the API call.

The generated responses are then parsed, and hallucinated references that are not part of the input are counted and removed from the final RAG output.

3 Submitted Runs and Code

Table 1 describes the variables we used in our implementation. In general, we submitted one baseline run (run # 1) without reranking and a simple variant of our RAG prompt. We experimented with different similarity measures in our runs (Cosine and Jaccard; found in runs # 1, 2, 3, 6 even though the prompts differ slightly), different weightings of similarity of segments to the query versus similarity of the segments to segments which have already been considered as relevant (through runs # 3, 4, 5, 6), different cutoff values for the number of segments considered by the generation step (through runs # 3, 7) and different prompts representing varying user knowledge on the query topic (through runs # 3, 8, 9). The priorities for evaluation are set to enable meaningful comparisons between parameter variations, even if not all runs are evaluated. A

detailed description of the properties of our ten submitted runs can be found in Table 3.

References

- [1] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Melbourne, Australia) (SIGIR '98). Association for Computing Machinery, New York, NY, USA, 335–336. <https://doi.org/10.1145/290941.291025>
- [2] Björn Engelmänn, Fabian Haak, Christin Katharina Kreutz, Narjes Nikzad-Khasmakhi, and Philipp Schaer. 2023. Text Simplification of Scientific Texts for Non-Expert Readers. In *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023)*, Thessaloniki, Greece, September 18th to 21st, 2023 (CEUR Workshop Proceedings, Vol. 3497). CEUR-WS.org, 2987–2998. <https://ceur-ws.org/Vol-3497/paper-250.pdf>
- [3] Björn Engelmänn, Christin Katharina Kreutz, Fabian Haak, and Philipp Schaer. 2024. ARTS: Assessing Readability & Text Simplicity. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, November 12–16*. Association for Computational Linguistics.
- [4] Christin Kreutz, Fabian Haak, Björn Engelmänn, and Philipp Schaer. 2024. BATS: Benchmarking Text Simplicity. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11–16, 2024*. Association for Computational Linguistics, 11968–11989. <https://doi.org/10.18653/v1/2024.FINDINGS-ACL.712>
- [5] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Trans. Assoc. Comput. Linguistics* 12 (2024), 157–173. https://doi.org/10.1162/TACL_A_00638

¹Note that these similarity values then can lie between -1 and 1, they need to be transformed to a non-negative range. <https://platform.openai.com/docs/guides/embeddings/embedding-models>

²snapshot gpt-4o-mini-2024-07-18