LLM-based Retrieval and Generation Pipelines for TREC Interactive Knowledge Assistance Track (iKAT) 2023

Zahra Abbasiantaeb
University of Amsterdam
Amsterdam, The Netherlands

Chuan Meng
University of Amsterdam
Amsterdam, The Netherlands
c.meng@uva.nl

David Rau
University of Amsterdam
Amsterdam, The Netherlands

Antonis Krasakis
University of Amsterdam
Amsterdam, The Netherlands

Hossein A. Rahmani
University College London
London, United Kingdom

Mohammad Aliannejadi
University of Amsterdam
Amsterdam, The Netherlands

ABSTRACT

The interactive Knowledge Assistant Track (iKAT) aims to develop personalized conversational assistants. In this task, the persona of the user is provided to the system before the conversation. iKAT consists of three main tasks including, Personal Textual Knowledge Base (PTKB) statement ranking, passage ranking, and response generation. We proposed two different pipelines to approach the task, namely, *retrieve-then-generate* and *generate-then-retrieve*. We submitted three runs based on the *retrieve-then-generate* pipeline using the Llama model and one run based on the *generate-then-retrieve* pipeline. The automatic run based on *generate-then-retrieve* pipeline outperformed the other automatic runs in the passage ranking task. This run achieved comparable results to the manual run based on the *retrieve-then-generate* pipeline. For the PTKB statement ranking task, we proposed two approaches including ranking PTKB statements using *(MiniLM12)* model and using the GPT-4 model as a zero-shot learner for classifying the PTKB statements as relevant or non-relevant. The ranking approach using *(MiniLM12)* model achieved better performance than the classification model approach.

1 INTRODUCTION

The TREC interactive Knowledge Assistant Track (iKAT) focuses on the task of personalized retrieval and answer generation, through conversational interfaces. The central objective of the iKAT competition is to design and develop interactive knowledge assistants capable of engaging users in dynamic and personalized dialogues, offering them comprehensive and contextually relevant information. Consider a user enquiring a conversational assistant for a complex information need, such as helping him/her decide on a Masters' degree. To tackle such complex information needs, assistants need to have access to personal information and preferences, conversational capabilities for elicitating and conveying information from/to the user as well as retrieval capabilities that would allow them to access and integrate relevant information from various sources.

This task expands upon the previous Conversational Assistant Tracks (CAsT) [3] in a number of ways. While both tracks require contextual information to answer the user's information need successfully, CAsT mostly requires context that originated from the conversation. iKAT adds a user personalization component to it, linking conversations to specific personas that embody certain characteristics or preferences (e.g., vegan; dietary preferences). Hence, the same user question should be answered differently depending on the personal information of the user. This is achieved by providing a Personal Textual Knowledge Base (PTKB), which defines various characteristics or preferences of a certain user (in Natural Language text). Based on that, the conversational agent has to retrieve personalized relevant information and answer the user question. In practice, that leads to different conversation trajectories that originate from the same question and evolve differently.

In addition to that, iKAT also allows for mixed-initiative interactions, such as the agent asking for a clarification question to the user.

Since a personalized assistant requires multiple reasoning, retrieval and language capabilities, the entire task of personalised retrieval and answer generation is broken down to multiple subtasks to allow for easier evaluation and understanding of the systems. Those are: (i) PTKB Provenance, (ii) Passage Provenance and (iii) Response Generation (explained further in Section 2)

Each topic includes several conversations of different users with different personas. The training topics consist of 11 conversations ranging over 8 topics, while the test topics contain 25 conversations over 13 topics. For the passage retrieval task, we rely on a document collection consisting of a subset of ClueWeb22-B, which was provided by the competition organizers.

We envision a pipeline approach, utilizing Large Language Models (LLM) and Information Retrieval (IR) systems to tackle the individual sub-tasks and provide grounded responses to the users' information needs (Section 3).

This paper is structured as follows: First, in §2 we give an overview of the iKAT tasks. In §3, we present our proposed approaches *retrieve then generate* and *generate then retrieve* in sub-sections. Next, in §4 detail our experimental setup. In §5 we present our results and discuss them. Finally, in §6 we conclude this work.

2 TASKS

This year iKAT provided *current user's information needs*, *conversation history*, and *Personal Text Knowledge Base (PTKB)* as the input

TREC'22, November 2022, Virtual

at each conversation turn to the participants and offered three different tasks according to these inputs, including:

(1) **PTKB Statement Ranking:** is a relevance score prediction task to determine the relevant statements from PTKB in each turn. The output of this task is a sorted list of the statements from PTKB with the relevance scores.
(2) **Passage Ranking:** aims to retrieve and rank the relevant passages from the given collection (i.e., user utterance, context, and PTKB) in response to a user statement.
(3) **Response Generation:** offers a set of text responses at each turn that can either be direct passages from the collection or generated summaries based on one or more passages. All responses must include at least one passage, referred to as "provenance", from the source collection.

Table 1: Retrieve then Generate pipeline LLM Instruction prompts for Query Rewriting and Response Generation with Llama 7B chat.

Query Rewriting:

### Instruction: 	
Rewrite the User Question to be self-contained, resolving references to the Conversational Context of all previous user questions and responses into account. Keep edits minimal.
 	
### Conversational Context: 	
{previous user utterance: u_1 }
{previous user utterance: … }
{previous user utterance: u_{t-1}}
{response: last system response (r_{t-1})}
 	
### User Question: 	
{current user utterance (u_t)}
 	
### Rewritten User Question: 	

Response Generation:

### Instruction: 	
Generate a Response to the Question. The answer should be informed by the User Background Information and be based on the Knowledge Base. Resolve references in the response.
 	
### User Background Information: 	
{PTKB statement$_{\text{top}-1}$}
{PTKB statement$_{\text{top}-2}$}
{PTKB statement$_{\text{top}-3}$}
 	
### Knowledge Base: 	
{Passage$_{\text{top}-1}$}
{Passage$_{\text{top}-2}$}
{Passage$_{\text{top}-3}$}
 	
### Question: 	
{rewritten user utterance (u'_t)}
 	
### Response: 	

3 METHODOLOGY

As previously outlined in Section 2, iKAT comprises three successive sub-tasks, with each sub-task functioning as a component in a pipeline. The components generate the input for the subsequent component. Consequently, the final answer is generated by aggregating the information from different components of the pipeline. For instance, PTKBs contain general background information about the user which may or may not be helpful to enrich the user utterance for passage (re-)ranking. To this end, first selecting relevant PTKBs can in the next step inform which passages are specifically relevant to the user and should be taken into account when generating the final response.

To solve the task, we have proposed two different pipelines namely *Retrieve-then-Generate* (**RtG**) and *Generate-then-Retrieve* (**GtR**); see Figure 1. Both pipelines employ generative LLMs to generate intermediate results where possible. In the first pipeline (*retrieve-then-generate*), we experiment with the trainable LLM (Llama 7B Chat), whereas for (*generate-then-retrieve*) we solely rely on the static GTP-4 API. We will explain each of these pipelines in the following.

3.1 Retrieve then Generate (RtG)

In our proposed *retrieve-then-generate* pipeline, we first aggregate relevant information through query rewriting, PTKB ranking, passage (re-)ranking, and finally use this information to generate a response. In more detail, first, the LLM is prompted to rewrite the user utterance given the current user utterance, the previous response of the system, and all previous user utterances. As a result, the rewritten query is representative of user information needs.

Next, the rewritten user utterance is used to obtain the most relevant PTKB statements. Selecting only relevant PTKB statements can be useful, as some PTKB statements can be unrelated to the user utterance. As a next step, the enriched user utterance and the selected PTKB statements are used to retrieve relevant passages in a multi-stage ranking. Finally, the rewritten query, relevant PTKB statements, and retrieved passages are used to generate the response to the user utterance.

We use Llama-7b-chat for all generation sub-tasks in this pipeline.

(1) **Query Rewriting *(Llama-7b-chat)*:**
We prompt the LLM to generate a self-contained rewritten query u'_t given the current user utterance u_t, the system response r_{t-1} at the previous turn, and all previous user utterances $\{u_1, \ldots, u_{t-1}\}$. The prompt used in this step is shown in Table 1.
(2) **PTKB Statement Ranking *(MiniLM12)*:**
In this step, we rank all PTKB statements using the rewritten user utterance u'_t as a query. We sort the PTKB statements according to a relevance score given by the neural ranking model. We empirically choose a cut-off threshold of 3, meaning we regard the top 3 PTKB statements as relevant and only select those for the next steps. The relevant PTKB statements for turn t are denoted as S_t which includes a subset of PTKB statements.

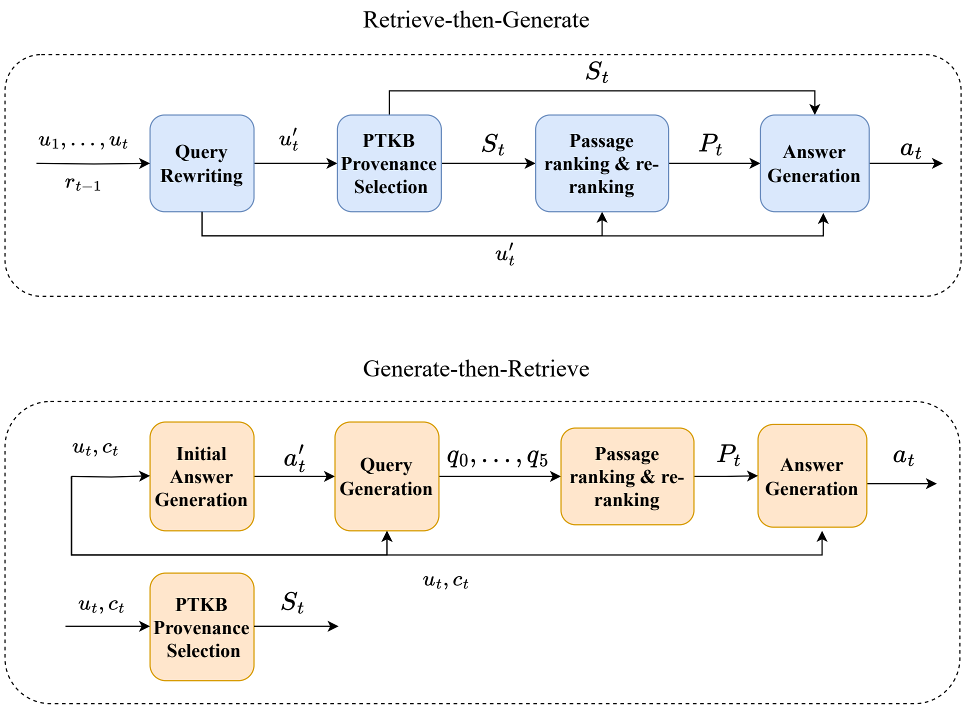


Figure 1: Overview of our two proposed pipelines including the Retrieve-then-Generate (RtG) and Generate-then-Retrieve (GtR).

(3) **Passage Retrieval *(BM25)*:**
We use the rewritten user utterance u'_t as a query (q_i) to rank all passages (D). Additionally, we expand the query with the selected PTKB statements from the previous step. The output of retrieval for each of the queries (q_i) is represented as $\{p_1^{(i)}, ..., p_n^{(i)}\}$. Top 100 passages for each query ($n = 100$) are selected and passed to the re-ranker.

(4) **Passage Re-Ranking *(MiniLM12)*:**
We re-rank the top 100 passages from BM25 using a state-of-the-art passage ranker to obtain a higher quality ranking. For this, we again expand the rewritten user utterance u'_t with the selected PTKB statements as a query. We sort the passages according to a relevance score given by the neural ranking model and similarly to the PTKB selection chose a cut-off threshold level of 3. The sorted list of relevant passages for turn t are represented as P_t.

(5) **Response Generation *(Llama-7b-chat)*:**
We prompt the LLM to generate the final response r_t given the generated rewritten query u'_t, top-3 selected PTKB statements, and the top-3 selected passages from the previous steps. The prompt used in this step is shown in Table 1.

We report a *Manual* and *Automatic* run, where the Automatic run uses outputs of previous steps in the pipeline as previously described. The Manual run considers the ground truth where possible e.g., for (2) PTKB selection the resolved utterance is used instead of the rewritten user utterance u'_t.

Note that for the steps (1) Query Rewriting and (2) Response Generation, we consider using the LLM zero-shot, and fine-tuned. As training data, we use the available ground truth resolved utterances, selected PTKB statements, and selected relevant passages from the train dataset.

3.2 Generate then Retrieve (GtR)

In this approach, the LLM is asked to generate an initial answer to the user utterance given the context of the conversation denoted as $c_t = \{(u_{t-1}, r_{t-1}), ..., (u_1, r_1)\}$ and the PTKB of the user. The LLM will generate the answer by doing reasoning over the context and PTKB of the user but the generated answer is not grounded to the documents within the collection. In the next step, the LLM is prompted to generate a set of 5 queries to achieve this answer. These generated queries are then passed to the first-stage retrieval model and re-ranker to retrieve and sort the relevant passages from collection. Each of the queries is passed to the first-stage retrieval and re-ranker resulting in a sorted list of 1000 passages for each query. The output of the re-ranker for each of the queries is interleaved to form a list of top 1000 relevant documents. To generate the final answer to the user utterance, top 2 passages for each query are passed to the LLM and the LLM is prompted to generate the answer based on the given passages, considering the context of the conversation and the PTKB of the user. To identify the PTKB

Table 2: Experimental results on the official evaluation set. Bold indicates the best results and *italic* shows the best results between Automatic runs.

Run	Pipeline	Submission	nDCG@3	nDCG@5	nDCG	P@20	Recall@20	mAP
`Llama zero-shot`	RtG	Automatic	0.1494	0.1437	0.0815	0.1165	0.0507	0.0387
`Llama fine-tuned`	RtG	Automatic	0.0826	0.0816	0.0457	0.0684	0.0301	0.0202
`GPT-4`	GtR	Automatic	**0.4382**	**0.4396**	**0.3479**	*0.3444*	*0.1821*	*0.1759*
`Llama fine-tuned`	RtsG	Manual	0.4122	0.4264	0.3245	**0.3530**	**0.2063**	**0.1911**

provenance statements, the LLM is prompted to select the relevant statements given the context of the conversation, current user utterance, and the PTKB. This task is solved as a binary classification problem. The statements identified as relevant are assigned the same score of relevance and non-relevant statements are assigned a zero score.

(1) **Initial Answer Generation *(GPT-4)*:**
The LLM is prompted to generate an initial answer (a'_t) to the user utterance (u_t). The prompt includes instructions for generating the answer, the context of the conversation (c_t) and the PTKB of the user (PTKB).
(2) **Query Generation *(GPT-4)*:**
After generating the initial answer (a'_t) by LLM in the previous step, the LLM is prompted in the same conversation to generate the required queries for retrieving the $a'_t$from D. In the prompt the model is asked to generate a maximum of 5 queries. The output of this step is a set of queries called $\{q_0, ..., q_m\}$ where $m < 5$.
(3) **PTKB statement ranking *(GPT-4)*:**
This task is approached as a binary classification problem. The LLM is given the instruction for selecting the relevant statements from PTKB to the current user utterance. The prompt includes the instruction, context of the conversation (c_t), PTKB of the user (PTKB), and current user utterance (u_t). The output of LLM is a list of relevant statements from PTKB shown as $\{s_i\}$.
(4) **Passage Retrieval *(BM25)*:**
Each of the generated queries (q_i) are passed to the BM25 ranker model. The output of retrieval for each of the queries (q_i) is represented as $\{p_1^{(i)}, ..., p_n^{(i)}\}$. The top 200 passages for each query ($n = 200$) are selected and passed to the re-ranker.
(5) **Passage Re-ranking *(MiniLM12)*:**
The first 200 passages retrieved for each query (q_i) are re-ranked using the pre-trained MiniLM12 model. The passage provenance is formed by interleaving the list of passages for each query and eliminating the duplicates from the list. Given m queries and m sorted list of passages like $\{p_1^{(i)}, ..., p_n^{(i)}\}$ for each query, the final passage provenance list is formed as $P_t = \{p_1^{(1)}, ..., p_1^{(m)}, p_2^{(i)}, ..., p_2^{(m)}, ..., p_n^{(1)}, ..., p_n^{(m)}\}$. The duplicate passages are removed from the passage provenance.
(6) **Response Generation *(GPT-4)*:**
The LLM is prompted to generate the answer to the user utterance from the top 10 passages. The prompt includes the instruction for generating the answer, the top 10 retrieved passages, persona of the user (PTKB), the context of the conversation (c_t), and the user utterance (u_t). The LLM generates the answer a_t to the user utterance (u_t).

4 EXPERIMENTAL SETUP

In this section, we detail the implementation details of the employed models.

(1) **BM25 Retriever:** As a first-stage retriever, we employ BM25 (default parameters) using Pyserini [2].
(2) **Neural Re-Ranker:** As a neural re-ranker we use the MiniLM12[1] as a zero-shot ranker with max. input length 512.
(3) **Llama LLM:** For our proposed *retrieve-then-generate* pipeline, we consider LLaMA [4] (Llama-2-7b-chat) as our LLM; specifically, we use Llama-2-7b-chat.[2] We consider LLaMA in a zero-shot or fine-tuned setting. For fine-tuning LLaMA, we use QLora [1] instead of updating the entire model. As for hyper-parameters of fine-tuning, we use a learning rate of 1e-4, set LoRA r as 64, set LoRA alpha as 16, set LoRA dropout as 0.1, set the batch size as 128, inject adapter layers to all linear layers, and train these adapter layers for 100 steps. For faster training and inference, we always use 4-bit quantization, half-precision. For generation, we set a maximum generation length of 128.
(4) **GPT-4 LLM:** The GPT-4 API[3] is used for the *generate-then-retrieve* pipeline.

5 RESULTS AND DISCUSSION

5.1 Overall performance

In this section, we discuss the official evaluation results of our four submitted runs. Table 2 shows the performances of our runs on different evaluation metrics. **Bold** indicates the best results of all runs and *italic* represents the best results between the Automatic submissions. As we can see from Table 2, as we expected, the `llama-fine-tuned (Manual)` run achieves the best performance among all models and by a large margin compared to `llama-zero-shot` and `llama-fine-tuned (Automatic)`. Nevertheless, it is evident that `GPT-4` significantly outperforms the Automatic runs, approaching performance levels similar to `llama-fine-tuned (Manual)`. This indicates the ability of GPT-4 for unseen tasks where it was highly effective in identifying the appropriate PTKB statements and generating efficient queries. Interestingly, comparing `llama-zero-shot` and *llama-fine-tuned* on Automatic submissions shows that `llama-zero-shot` outperforms `llama-fine-tuned` on all evaluation metrics. In preliminary experiments with Llama, we found ending the prompt instruction with a specific instruction such as

[1] https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2
[2] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[3] https://chat.openai.com/

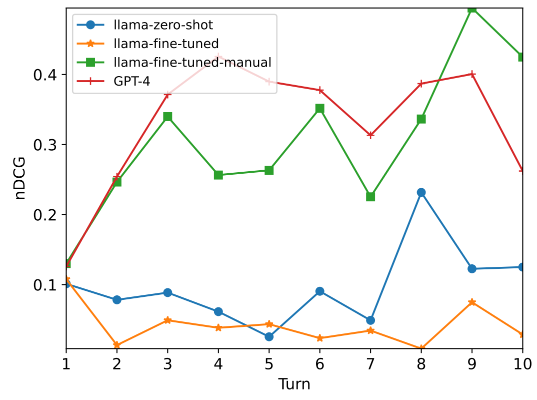
Table 3: Generate then Retrieve pipeline LLM Instruction prompts for (1) Initial Answer Generation, (2) Query Generation, and (3) Response Generation using GPT-4.

(1) Initial Answer Generation and (2) Query Generation.
Instruction: *I will give you a conversation between a user and a system. Also, I will give you some background information about the user. You should answer the last utterance of the user. Please remember that your answer to the last question of the user shouldn't be more than 200 words.* ### Background information about the user: PTKB statements ### Conversation: ### $(u_1, r_1), ..., (u_{t-1}, r_{t-1}), (u_t)$ ### Response: ... ### Can you generate the unique queries that can be used for retrieving your previous answer to the user? (Please write each query in one line and don't generate more than 5 queries) ### Queries: ...
(3) Response Generation:
Document 1: ... ### .... ### Document 10: ... ### Instruction: *I will give you a conversation between a user and a system. Also, I will give you some background information about the user. You should answer the last utterance of the user by providing a summary of the relevant parts of the given documents. Please remember that your answer shouldn't be more than 200 words.* ### Background information about the user: PTKB statements ### Conversation: ### $(u_1, r_1), ..., (u_{t-1}, r_{t-1}), (u_t)$ ### Response: ...

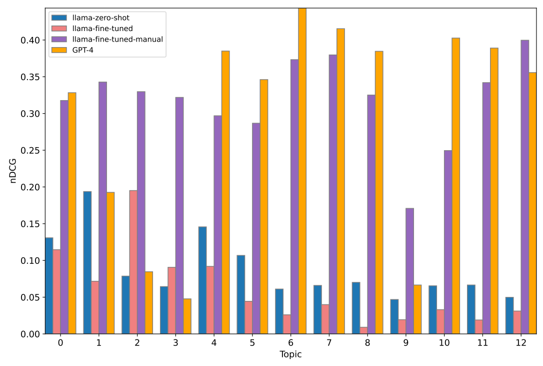
"Rewritten User Question:" to be important. We found that ending the instruction prompt with "Answer:" would misguide the model to directly answering the user utterance ignoring the provided context. Further, we observed that using more descriptive terms such as "Background Information", and "User Question" yields better results. Table 4 also shows the performance on PTKB selection. For PTKB selection we used two different models, `Sentence Transformer` and `GPT-4`, as one can see, `Sentence Transformer` achieved better performance in selecting more relevant PTKB than `GPT-4`.

Table 4: Experimental results on the PTKB selection. Bold indicates the best results.

Run	nDCG@5	P@5	Recall@5	mAP
`Sentence Transformer`	**0.7645**	**0.3592**	**0.8320**	**0.7448**
`GPT-4`	0.6524	0.2522	0.6608	0.5799



(a) Results per turn



(b) Results per Topic.

Figure 2: Results per conversations and topics

5.2 Performance per conversation and topic

Figure 2 shows the performance of our runs averaged per conversations and topics in terms of nDCG. Both plots show that `GPT-4` and `llama-fine-tuned (Manual)` achieve significant performance on all conversations and topics. `llama-fine-tuned (Manual)` uses the manually annotated data of the resolved utterances and PTKB provenance statements, therefore, it was expected to show a high and robust performance. Interestingly, `GPT-4` has shown a very comparable performance for each conversation and topic with the `llama-fine-tuned (Manual)` even though it is an automatic run without any manually annotated data. However, we observe a gradual decrease in performance towards later conversations and topics (e.g., conversation 19 in Figure 2a and topic 10 in Figure 2b) which need further investigation).

6 CONCLUSION

In this paper, we describe our submissions in TREC iKAT 2023. We proposed two different pipeline methodologies to tackle the iKAT tasks, namely, *Retrieve-then-Generate* and *Generate-then-Retrieve*. Using Llama as zero-shot learner performed better than fine-tuning the Llama for the task. The GPT-4 model was used in the GtR pipeline. Using GPT-4 in and the GtR pipeline outperformed the other manual runs, achieving a comparable result to the Automatic run. In addition, using neural ranking model for the PTKB statement ranking task performed better than using GPT-4 for classifying the PTKB statements as relevant or non-relevant.

REFERENCES

[1] Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: Qlora: Efficient finetuning of quantized llms. arXiv preprint arXiv:2305.14314 (2023)

[2] Lin, J., Ma, X., Lin, S.C., Yang, J.H., Pradeep, R., Nogueira, R.: Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In: Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021). pp. 2356–2362 (2021)

[3] Owoicho, P., Dalton, J., Aliannejadi, M., Azzopardi, L., Trippas, J.R., Vakulenko, S.: Trec cast 2022: going beyond user ask and system retrieve with initiative and response generation. In: Text REtrieval Conference Conference (2022)

[4] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)