What to Ask LLM: Knowledge-aware Follow-up Query Generation for Progressive Information Acquisition

Anonymous COLING 2025 submission

Abstract

The advent of Large Language Models (LLMs) has significantly eased the process of knowledge acquisition for users through interactive question answering. However, when confronted with domain-specific knowledge, users may struggle to formulate relevant questions aligned with the initial query. To address this challenge, in this paper, we introduce a novel task termed **K**nowledge-aware Follow-up **Q**uery **G**eneration (KQG), which aims to generate a sequence of follow-up queries with diverse knowledge to aid users in progressive knowledge acquisition from LLMs. To facilitate this task, we first construct a new dataset tailored specifically for KQG, sourced from an online knowledge sharing community. Subsequently, we propose a novel end-to-end training framework for KQG, named ReSAG (**Re**trieval-then-**S**electing **A**ugmented **G**eneration), which extends typical Retrieval-Augmented Generation (RAG) methods by incorporating a selecting policy network. To be specific, ReSAG comprises three main components: a knowledge retriever, a selecting policy network, and a T5-based question generator. To train our framework in an end-to-end manner, we introduce a novel variant of policy optimization that integrates neural dense retrieval and selecting into a T5-based sequence-to-sequence generation model, using only ground-truth target output. Finally, extensive experiments demonstrate that our approach significantly outperforms existing state-of-the-art methods, including generative models, RAG models, and LLM-based methods.

1 Introduction

Recent advancements in Large Language Models (LLMs), including OpenAI's GPT-3.5 (Ouyang et al., 2022), GPT-4 (Achiam et al., 2023), Google's PaLM (Chowdhery et al., 2023), ChatGLM (Zeng et al., 2022), and other benchmark models (Touvron et al., 2023; Taori et al., 2023; Chiang et al., 2023), have showcased remarkable capabilities in knowledge-oriented question answering. These models greatly facilitate the knowledge-seeking process for users through interactive question answering. However, challenges arise when dealing with domain-specific knowledge, as users may struggle to pose relevant queries to acquire more information. This difficulty often stems from a lack of awareness regarding knowledge relations and taxonomy. As illustrated in Figure 1, consider the case of a student querying information about *singleton design pattern in C++*. Despite the initial inquiry, the user may fail to acquire further knowledge about related topics such as *lazy initialization* and *eager initialization*, which are integral to understanding *the singleton design pattern*.

To address the aforementioned challenges, one seemingly straightforward approach is to prompt LLMs directly to generate the follow-up queries. Figure 1 illustrates an example where a LLM produces queries based on such a prompt. Although each suggested query is relevant to the initial query, they often lean towards superficial queries of the "What", "How", and "Which" types, lacking depth in knowledge exploration. This tendency overlooks the intricate relationships within knowledge, thereby limiting their effectiveness in aiding user knowledge acquisition. The above phenomenon can be attributed to two main reasons: Firstly, simplistic prompt designs fail to elicit professional and complex knowledge relations from LLMs, resulting in discontented query generation. Secondly, LLMs often exhibit limited performance in domain-specific knowledge-oriented reasoning tasks (Wang et al., 2023), as they struggle to capture the nuanced structure of knowledge during generation. While fine-tuning and maintaining LLMs to integrate domain-specific knowledge may alleviate this issue to some extent, it can pose a significant financial burden for many institutions and researchers. Additionally, the limited availability and confidentiality of domain-specific data introduce risks of

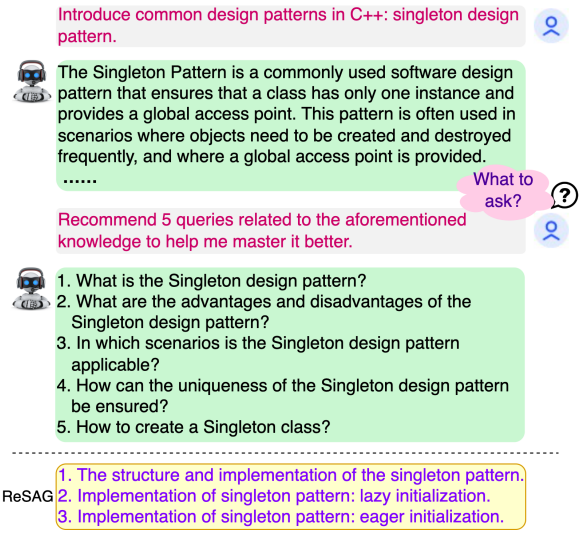

Figure 1: An illustrating example of follow-up query generation for a LLM-based prompting method and our ReSAG.

potential data leaks.

To enhance the performance of LLMs on domain-specific reasoning tasks, recent research efforts (Shi et al., 2023; Peng et al., 2023) have primarily focused on employing a held-out dense retriever to extract knowledge and then leveraging that knowledge for generating results. This approach, known as retrieval-augmented generation (RAG), has demonstrated success in various knowledge-intensive tasks. However, these methods often retrieve homogeneous knowledge with a high degree of similarity, posing a challenge for the model to differentiate between similar information (Shen et al., 2023). Consequently, this may lead to the inadvertent selection of inappropriate retrieved evidences, failing to produce the diverse knowledge. Along this line, in this paper, we introduce a novel task called **K**nowledge-aware Follow-up **Q**uery **G**eneration (KQG), aimed at producing a sequence of queries with diverse knowledge based on an initial query. These queries serve as a progressive means of knowledge acquisition from LLMs for users. To facilitate this, we construct a dataset specifically tailored for KQG from an online knowledge sharing community focusing on information technology. This dataset comprises a large collection of query and follow-up queries pairs characterized by high correlation, intended for academic research purposes.

Building upon this dataset, we propose a novel end-to-end training framework for KQG, named ReSAG (**Re**trieval-then-**S**electing **A**ugmented **G**eneration), which extends typical Retrieval-Augmented Generation (RAG) methods by incorporating a selecting policy network. This network connects the generation and retrieval processes, learning an optimal policy to select the suitable evidences from the retrieved passages so that it facilitates the generator in producing multiple queries with associated knowledge, while also diversifying the retrieved evidences to prevent the homogeneity of generation results. The proposed ReSAG consists of three main components: a knowledge retriever, a selecting policy network, and a T5-based query generator. To train our framework in an end-to-end manner, we introduce a novel variation of policy optimization, which allows us to train the initial retriever, policy networks, and generator using only ground-truth target output. Our main contributions are summarized as follows:

(1) We introduce a novel task and dataset, KQG, designed to generate a sequence of associated knowledge-aware follow-up queries based on the initial query by retrieving evidence passages from a knowledge base. This large-scale KQG dataset comprises 1,677,513 document passages for evidence retrieval and 210,531 (query, follow-up queries) pairs for model training, validation, and testing purposes [1].

(2) We propose an end-to-end training framework for KQG, which integrates neural dense retrieval and selecting process into a T5-based sequence-to-sequence generation model without the need for query-passage matching labels.

(3) Our approach achieves superior performance compared to existing state-of-the-art methods, including generative models, RAG models, and LLM-based methods, by a significant margin.

2 Related Works

Incorporating external knowledge into parametric language models for text generation has become a focal point for researchers. Leveraging the effectiveness of the dual-encoder neural retriever (Karpukhin et al., 2020), the RAG framework has seen extensive adoption in various text generation tasks. This framework utilizes a dense retriever to fetch passages from a document library, using the retrieved content as input for a generator to formulate responses. Notable implementations include RAG (Lewis et al., 2020), which encodes each retrieved pasages alongside the query and computes the probabilities of the answer. FiD (Izacard and Grave, 2020b) adopts a similar approach to RAG, encoding each retrieved

[1]The details of data collection and processing can be found in Appendix A.1.

evidence passages and amalgamating their hidden states in the decoder. FiD-KD (Izacard and Grave, 2020a) and EMDR2 (Singh et al., 2021) are variations based on FiD, differing in retriever training methods: FiD-KD utilizes knowledge distillation, whereas EMDR2 leverages marginal likelihood. REPLUG (Shi et al., 2023) extends RAG to large language models but updates only the retriever during training. Unlike KQG, these RAG methods primarily retrieve homogeneous information for single-target generation. Re^{2}G (Glass et al., 2022) bears the closest resemblance to our work, focusing on retrieving and reranking for sequence-to-sequence generation. However, its initial retrieval and reranking phases necessitate the use of provenance ground truth, which is not feasible for our KQG. Departing from the conventional ranking paradigm, ReSAG views the ranking process as a sequential decision-making process, enabling consideration of both knowledge relevance and diversity within our KQG framework.

3 Task Definition

In this section, we formally define the task of KQG and introduce the necessary notation. KQG aims to predict a list of follow-up queries, denoted as $Y = [Y_1, Y_2, ...]$, based on the given initial query X (X and Y are from the training corpus $\mathcal{D}$). This process serves to facilitate progressive knowledge acquisition from a Large Language Model (LLM) denoted as F_{lm}. Directly generating follow-up queries from the initial query X proves challenging due to limited information within X [2]. Additionally, we introduce a document-based knowledge base $\mathcal{Z}$, comprising a large-scale collection of document passages for evidence retrieval to support the knowledge-oriented query generation. It is worth noting that the sets $\mathcal{D}$ and $\mathcal{Z}$ are non-parallel, indicating that each X is associated with a specific Y. However, the relationship between a document Z in $\mathcal{Z}$ and the tuple (X, Y) is not explicitly defined. In addition, our KQG differs from typical RAG tasks in several ways. While RAG tasks utilize the input query to retrieve text documents for additional context during target sequence generation, our KQG differs in two fundamental aspects:

(1) Unlike RAG tasks, which focus on a single target sequence, KQG is tailored for the generation of multiple target sequences.

[2]Actually, we utilize F_{lm} to produce the response based on X and concatenate X with its response as the final input for the subsequent retrieval and generation process.

(2) RAG tasks typically retrieve multiple homogeneous documents as evidences to facilitate the generation of a single target sequence. In contrast, KQG selects diverse evidences to support the generation of varied queries with diverse knowledge.

4 Methodology

In this section, we present our proposed framework, i.e., ReSAG, designed to address the KQG task. An overview architecture of the model is presented in Figure 2. ReSAG comprises three pivotal components: 1) a dense knowledge retriever, 2) a query generator, and 3) a selecting policy network. Different from the RAG framework, we incorporate an additional selecting policy network to bridge the gap between the generation and retrieval processes. This integration enhances the awareness of associated knowledge in the process of query generation while also promoting diversity in the retrieved evidences, thus mitigating the risk of homogeneous generation results. Specifically, the retriever P_η retrieves the top-N passages $\tilde{\mathcal{Z}} = \{Z_1, \cdots, Z_N\}$ based on $\mathcal{Z}$ and X. Then, the selecting policy network P_ϕ learns to select the optimal K evidences to obtain top-K passages $\hat{\mathcal{Z}} = \{Z_1, ..., z_K\}$ by sequential decision learning, where $K \leq N$. It is worth noting that we use a larger evidence space N to explore more potential evidences beneficial for query generation. Finally, the generator P_θ produces the target query sequence based on the top-K selected passages and the query X. In reference to the objective of RAG outlined in Equation 18, we formulate ReSAG as follows:

$$\mathcal{L}(\eta, \phi, \theta) = -log \sum_{Z_i \in \hat{\mathcal{Z}}} P_\theta(Y|X, Z_i) P_\phi(\hat{\mathcal{Z}}|\tilde{\mathcal{Z}}) P_\eta(\tilde{\mathcal{Z}}|X), \tag{1}$$

where η, ϕ and θ are the parameters of the retriever, the selector and the generator, respectively.

Within this framework, we introduce a novel variation of policy optimization, which allows us to train the retriever P_η, policy network P_ϕ, and generator P_θ using only ground-truth target output Y. In the subsequent sections, we formally introduce three modules and elucidate the joint training procedure.

4.1 Knowledge Retriever

Let the set of evidence passages be represented by $\mathcal{Z} = \{Z_1, Z_2, ..., Z_{|\mathcal{Z}|}\}$. Given a query X, the objective of the retriever module is to identify a subset of evidence passages $Z \in \mathcal{Z}$ as additional context that can effectively facilitate the corresponding follow-up query generation. The retrieval module, denoted

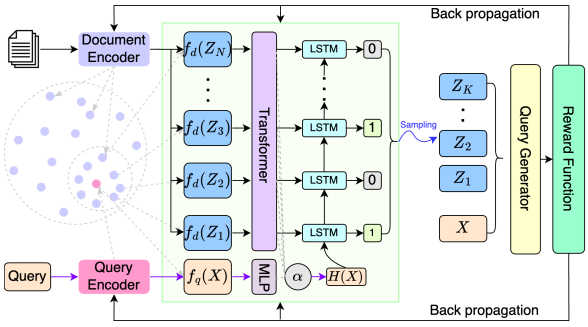

Figure 2: The overview architecture of our ReSAG.

as $P_\eta(Z|X)$, is constructed based on the conventional dense passage retriever (DPR) (Karpukhin et al., 2020). DPR employs a dual encoder architecture (Yih et al., 2011) wherein one encoder, f_q, encodes the initial query, and another, f_d, encodes the evidence passage into a vector. We leverage transformer encoders for f_q and f_d, employing an architecture similar to BERT (Kenton and Toutanova, 2019) with 12 layers and a hidden size of 768. The ultimate representation of the initial token (i.e., the standard [CLS] token from BERT's tokenization) serves as our query and passage embedding. It's worth noting that initializing f_q and f_d with a pretrained BERT weights and then pretraining with contrastive learning in the SimCSE model [9] have been proven to avoid cold-starting, resulting in normal retrieval accuracy.

$$\begin{aligned} f_q(X) &= BERT_q(X) \\ f_d(Z) &= BERT_d(Z) \\ score(X,Z) &= f_q(X)^T f_d(Z) \\ P_\eta(Z|X) &\propto exp(score(X,Z)) \end{aligned} \tag{2}$$

where $f_q(X)$ denotes a query representation generated by a dedicated query encoder, while $f_d(Z)$ represents a dense passage representation produced by a passage encoder. The retrieval score, denoted as $score(\cdot,\cdot)$, is determined by the dot product between the two resultant vectors.

For the computation of top-N passages, top-$N(P_\eta(Z|X))$, where N passages $\tilde{\mathcal{Z}}$ exhibit the highest prior probability $P_\eta(Z|X)$, poses a Maximum Inner Product Search (MIPS) (Shrivastava and Li, 2014) challenge. This problem can be effectively addressed through an approximate solution in sub-linear time. To achieve this, the passage embedding vectors are pre-calculated and organized into an index using Locality Sensitivity Hashing (LSH) (Datar et al., 2004). This indexing allows the query vector to be hashed to a cluster containing passages that are relatively pertinent. While this search strategy is approximate, it demonstrates compelling empirical search results, particularly when dealing with extensive passage sets. We denote the set of retrieved passages as $\tilde{\mathcal{Z}} = \{Z_1, \cdots, Z_N\}$.

4.2 Selecting Policy Network

The retriever's tendency to retrieve homogeneous evidence poses a challenge for generating effective follow-up queries. To overcome this limitation, we elaborately devise a selecting policy network, which is designed to learn an optimal policy for the evidence selection, enabling the generation of follow-up queries with diverse knowledge. Thus, this process is conceptualized as a sequential decision-making process. The selecting policy network mainly consists of three modules. Based on the retrieved top-N evidence passages [$f_d(Z_1)$, $f_d(Z_2)$, ..., $f_d(Z_N)$], a context module first captures contextual-aware information for the retrieved passages. Subsequently, an aggregation module enhances the query representation by incorporating this contextual-aware information, enabling the capture of correlations between the query and the retrieved passages. Finally, a evidence selection module is employed to learn and select these passages, obtaining the optimal top-K evidences to maximize the evaluation metrics of the generated queries.

Context module. It is evident that the retrieved passages are contextually dependent, prompting the application of a multi-layer Transformer network to encode contextual-aware information for the evidence passages. Formally,

$$[H_{Z_1}, ..., H_{Z_N}] = \text{Transformer}([f_d(Z_1), ..., f_d(Z_N)]) \tag{3}$$

where $f_d(Z_i)$ denotes the representation of the i-th passage obtained from the retriever in Eq. 2, and H_{Z_i} represents the corresponding context-aware passage representation.

Aggregation module. This module serves to amalgamate the original query representations within the evidence passages, yielding an aggregated query representation. An attention network is applied in this module, defined as follows:

$$\begin{aligned} H_X &= \sum_{i=0}^{N} \alpha_i * H_{Z_i} \\ \alpha_i &= \frac{\exp(\text{MLP}([f_q(X); H_{Z_i}])}{\sum_{j=0}^{N} \exp(\text{MLP}([f_q(X); H_{Z_j}])} \end{aligned} \tag{4}$$

where H_X represents the aggregated query representation. The term MLP$(\cdot)$ signifies a two-layer fully connected network utilizing the rectified linear unit as the activation function. The notation $[\cdot;\cdot]$ denotes the concatenation of two vectors.

Evidence selection module. This module sequentially assigns labels of 1 (indicating relevance to a target output) or 0 (otherwise) to each evidence passage. Implemented with a LSTM (Hochreiter and Schmidhuber, 1997) network and a softmax layer, the module takes the passage H_{Z_i} and the query H_X as input at time t_i and makes a binary prediction, conditioned on contextual-aware evidence passage representation and the previously labeled passages. This process discerns both locally and globally important passages, thereby supporting diverse query generation. The passages are selected based on $P(A_i = 1|Z_i)$, the confidence scores assigned by the softmax layer of the policy network.

$$P_\phi(A_i|X, Z_i) = \text{softmax}(\text{LSTM}(H_{Z_i}, H_X)). \quad (5)$$

The learning process involves selecting passages through training our network within a reinforcement learning framework. This entails the direct optimization of final evaluation metrics, specifically ROUGE (ROUGE, 2004) and BLEU (Papineni et al., 2002) in our query generation.

4.3 Query Generator

After the evidence selection, we obtain the top-K passages $\hat{\mathcal{Z}}$. The query generator takes a query X and $\hat{\mathcal{Z}}$ as input to produce the target queries Y. In our KQG setting, we encourage the retriever to recall diverse passages, facilitating the generation of a variety of target queries. Consequently, following prior RAG tasks, we employ Fusion-in-Decoder model (Izacard and Grave, 2020b) as the generator, which can fuse pertinent information from the top-K retrieved passages. Specifically, the query generator is built on top of T5 (Raffel et al., 2020), a pre-trained sequence-to-sequence transformer featuring an encoder g_e and a decoder g_d. For the encoder's input, each retrieved passage $Z_k \in \hat{\mathcal{Z}}$ is initially appended with the query:

$$I_k = [CLS]\, X\, [SEP]\, Z_k\, [SEP], \quad (6)$$

where "[CLS]" marks the beginning of a passage, and "[SEP]" serves as a separator for distinct parts of the passage as well as the final token. Each I_k is then independently fed as input to the T5 encoder g_e. The resulting representations corresponding to all retrieved passages are concatenated as follows:

$$I_{1\sim K} = [g_e(I_1), ..., g_e(I_K)] \in \mathcal{R}^{T\times K\times D} \quad (7)$$

where T represents the number of tokens in each I_k, and D is the hidden size of the T5 encoder g_e. Subsequently, $I_{1\sim K}$ serves as input to the T5 decoder g_d. Similar to the paradigm of text generation, during the generation of a target token, the decoder generate each token in the target output with both causal attention and cross-attention. In other words, it produce current token by incorporating both previously generated token and the tokens encoded in $I_{1\sim K}$. Since $I_{1\sim K}$ encompasses information from multiple passages, the decoder can aggregate valuable information from various sources and engage in joint reasoning.

Finally, we define the probability of the generated queries as:

$$P(Y|X, \tilde{\mathcal{Z}}; \theta) = \prod_{t=1}^{l} P_\theta(y_t|y_{<t}, X, \hat{\mathcal{Z}}; \phi), \quad (8)$$

where θ denotes the generator parameters and l is the number of generated query tokens. The generation of query tokens continues until the decoder outputs a special [EOS] token or the generated queries reaches a pre-specified maximum length.

4.4 Training via Reinforcement Learning

In this section, we extend the application of reinforcement learning to bridge the retrieval and generation processes. Specifically, we formulate an objective function, or reward function, to assess how effectively the retrieved passages contribute to the generation of the target queries. This reward function is utilized to globally optimize the evaluation metrics of query generation, namely ROUGE (ROUGE, 2004) and BLEU (Papineni et al., 2002), through policy gradient reinforcement learning. Our training algorithm facilitates exploration within the space of potential evidence, supporting follow-up query generation with diverse knowledge. Consequently, reinforcement learning contributes to KQG in two key ways: (a) by directly optimizing the evaluation metrics instead of maximizing the likelihood of reference target queries and (b) by enhancing our model's ability to discriminate among evidence passages. A passage is selected if it effectively supports the generator in producing high-scoring target queries.

Training procedure. We train a policy π_ψ with parameters ψ to predict binary labels, where $\psi = \{\eta, \phi\}$. Given the retrieved top-N passages $\tilde{\mathcal{Z}} = [Z_1, Z_2, ..., Z_N]$, the policy π_ψ generates a binary probability distribution for each Z_i in $\tilde{\mathcal{Z}}$. We denote the collection of these distributions for all passages in $\tilde{\mathcal{Z}}$ as $\pi_\psi(*|X, \tilde{\mathcal{Z}})$. The probability $\pi_\psi(A|X, \tilde{\mathcal{Z}})$ of an action sequence A can be formulated as follows:

$$\pi_\psi(A|X, \tilde{\mathcal{Z}}) = \prod_i P_\phi(A_i|X, Z_i) \quad (9)$$

where $\pi_\psi(A|X, \tilde{\mathcal{Z}})$ represents the probability of including a passage Z_i when $A_i = 1$ or excluding it when $A_i = 0$. When selecting evidences using π_ψ, we choose the label with the higher score for each passage:

$$A^b = [\underset{A_i}{argmax}\, \pi_\psi(A_i|X, Z_i)\ for\ Z_i \in \tilde{\mathcal{Z}}] \quad (10)$$

We employ a policy gradient technique (Sutton et al., 1999) to train our model. Unlike conventional sequential reinforcement learning setups, our π_ψ takes only one action for a given input, promptly receiving the corresponding reward without transitioning through other intermediate states. In essence, a single action corresponds to a specific label sequence $A = (A_1, A_2, ..., A_N) \in \{0, 1\}$). However, in our scenario, the policy generally has the flexibility to access rewards for multiple potential actions through sampling.

The training objective aims to maximize the expected reward assigned to a predicted label sequence A given a set of input retrieved passages $\tilde{\mathcal{Z}}$, as computed by the reward function R:

$$\mathcal{J}(\psi) = \mathbf{E}[R(\tilde{\mathcal{Z}}, A)]. \quad (11)$$

Then, according to he policy gradient theorem (Sutton et al., 1999), the gradient of this expectation can be expressed as follows:

$$\nabla_\psi \mathcal{J}(\psi) = \nabla_\psi \mathbf{E}[R(\tilde{\mathcal{Z}}, A) \log \pi_\psi(A|X, \tilde{\mathcal{Z}})] \quad (12)$$

Given the intractability of computing this expectation for a large dataset and the corresponding action space, the gradient is estimated through sampling:

$$\nabla_\psi \mathcal{J}(\psi) = \nabla_\psi r^s \log \pi_\psi(A^s|X, \tilde{\mathcal{Z}})] \quad (13)$$

where $A^s \sim \pi_\psi(\cdot|X, \tilde{\mathcal{Z}})$ represents a sample from the current policy at a given step, comprising binary passage labels $A^s = (A_1^s, A_2^s, ..., A_N^s)$, and $r^s = R(\tilde{\mathcal{Z}}, A^s)$.

In line with common practice in policy gradients, we introduce a baseline subtraction from the reward to reduce variance. We define the baseline as $r^b = R(\tilde{\mathcal{Z}}, A^b)$, representing the reward assigned to the most likely label sequence A according to the current policy. The resulting gradient is expressed as:

$$\nabla_\psi \mathcal{J}(\psi) = \nabla_\psi (r^s - r^b) \log \pi_\psi(A^s|X, \tilde{\mathcal{Z}})] \quad (14)$$

Consequently, our model is trained by minimizing the following loss function:

$$\mathcal{L}_\psi = (r^b - r^s) \log \pi_\psi(A^s|X, \tilde{\mathcal{Z}})]. \quad (15)$$

The incorporation of the baseline r^b facilitates an intuitive interpretation: a sample A^s is encouraged if its reward surpasses the current policy's prediction (i.e., when the factor $r^b - r^s$ is negative), and discouraged otherwise.

Reward Function. The reward function is intended to ensure the connection of retriever and generation process. The selecting policy network perform the action to select K evidence passages from $\tilde{\mathcal{Z}}$. Then, the generator produce the corresponding queries $\hat{Y}$ based on the selected K passages. We use evaluation metrics, namely ROUGE-1, ROUGE-2, ROUGE-L and BLEU-4 to calculate reward as follows:

$$R(\tilde{\mathcal{Z}}, A) = \lambda_1 * \text{ROUGE-1}(Y, \hat{Y}) + \lambda_2 * \text{ROUGE-2}(Y, \hat{Y}) + \lambda_3 * \text{ROUGE-L}(Y, \hat{Y}) + \lambda_4 * \text{BLEU-4}(Y, \hat{Y}) \quad (16)$$

where the evaluation scores ROUGE-$*(Y, \hat{Y})$ and BLEU-4$(Y, \hat{Y})$ quantify the performance of the model. The weights λ_1, λ_2, λ_3, and λ_4 serve as hyperparameters for each metric, and it is important to note that $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$ equals 1.0. Within this reward function, ROUGE-1 and ROUGE-2 measure unigram and bigram overlap, providing an assessment of informativeness, while the longest common subsequence (ROUGE-L) is employed to evaluate fluency. BLEU-4 is used to evaluate the similarity between a generated text and a reference text. The whole training process for ReSAG is shown in Algorithm 1 in Appendix A.7.

5 Experiments

5.1 Evaluation Metrics

In contrast to previous works (Du et al., 2017; Yao et al., 2018) on query generation evaluation that measure the quality of a generated query with respect to a single reference query using widely adopted similarity metrics such as BLEU (Papineni et al., 2002) and ROUGE (ROUGE, 2004), our ReSAG generates a list of queries, and there are multiple gold reference queries. To maintain consistency with single-target evaluation, we concatenate all generated queries and the gold reference queries separately. Subsequently, we employ metrics, including BLEU-4, ROUGE, and BERTScore (Zhang et al., 2019) and Diversity (Li et al., 2015), to conduct evaluations from the perspectives of lexical match, contextual embedding semantics and knowledge diversity. The detailed description for these mertics can be found in Appendix A.4.

5.2 Baseline Methods

To verify the effectiveness of our ReSAG, we select several baselines, including generative approaches (i.e., GPT2 and T5), large language model

Table 1: Main results on validation and testing datasets.

Approaches	Validation Dataset					Testing Dataset				
	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-4	BERTScore	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-4	BERTScore
GPT2	13.482	2.052	10.348	1.488	0.589	13.439	2.037	10.287	1.485	0.590
T5	22.406	6.686	17.406	2.187	0.674	22.300	6.761	17.234	2.183	0.672
Prompt4LLM	15.126	2.694	11.516	1.193	0.613	14.999	2.673	11.396	1.178	0.611
ICL	16.585	3.36	12.649	1.551	0.622	16.120	3.037	12.374	1.34	0.620
P-Tuning	20.86	5.48	13.9	1.520	0.610	20.410	5.46	13.41	1.433	0.608
LoraFtuneLLM	21.464	6.015	16.340	2.025	0.639	21.751	6.158	16.553	2.062	0.661
FID-KD	23.902	7.232	18.504	3.103	0.681	23.827	7.139	18.334	3.005	0.679
RetGen	24.632	8.038	19.267	3.955	0.692	24.428	7.933	19.142	3.904	0.681
ReSAG	**28.633**	**13.555**	**22.332**	**7.843**	**0.706**	**28.701**	**13.568**	**22.167**	**7.728**	**0.706**
- w/ DQN	26.595	11.156	20.49	6.132	0.664	26.613	11.239	20.444	6.214	0.668
- w/o SPN	21.663	6.843	16.579	2.773	0.425	21.974	6.904	16.619	2.811	0.432
- w/o DPR-P	25.009	10.37	19.383	5.132	0.554	25.182	10.281	19.258	5.091	0.532

based approaches (i.e., P-Tuning, LoraFtuneLLM, Prompt4LLM and ICL) and typical RAG method (i.e., FID-KD). The detailed description for these baselines can be found in Appendix A.3.

In our baselines, for LoraFtuneLLM, Prompt4LLM and ICL, we select Baichun2 (Baichuan, 2023) as our testing LLM. In addition, we also introduce some variants of our ReSAG:

- **ReSAG w DQN** uses deep Q-learning network (Wan et al., 2021) to substitute policy gradient algorithm for the policy learning.
- **ReSAG w/o SPN** is variants of our ReSAG without using selecting policy network to bridge the retrieval and generation process. Thus, our ReSAG degrades to a retrieve-then-generate model.
- **ReSAG w/o DPR-P** is variants of our ReSAG, in which the retriever is only initialized with a pre-trained Chinese BERT model, without pre-training using contrastive learning technique in the SimCSE model (Gao et al., 2021).

5.3 Results

5.3.1 Main Results

The overall results of ReSAG and all baselines on validation and testing datasets are reported in Table 1. The experimental results show that ReSAG achieves the best performance on all metrics, including ROUGE-1, ROUGE-2, ROUGE-L and BLEU-4 and BERTScore, demonstrating its effectiveness. On these metrics, we observe that ReSAG obtains an improvement of 4.001%, 5.517%, 3.065%, 3.888%, 0.014 over the best baseline (i.e., RetGen) on the validation dataset and 4.273%, 5.635%, 3.025%, 3.824%, 0.025 on the testing dataset. It is worth noting that all baselines do not perform very well. There may exist the following several reasons: GPT2 and T5 are typical generative models. It is difficult for them to only use the initial query as source information to produce the knowledge-aware follow-up queries. It is also hard to capture the inherent relations in knowledge. Prompt4LLM and ICL are unsupervised LLM-based prompting approaches. Since LLM has its inherent generative styles, misalignment between the generated results from LLM and the reference results lead to a bad performance. Although P-Tuning and LoraFtuneLLM are supervised LLM-based approaches, fine-tuning on the training dataset, they also perform not good because it is hard to elicit the knowledge relations from LLMs in our KQG. For FID-KD, it a typical RAG model, it perform worse in our multiple target generation scenario. While both our approach and RetGen leverage signals from the generation process to optimize the retrieval process, ReSAG intricately designs a novel sequential decision-making learning algorithm within the selection layer, resulting in enhanced performance.

5.3.2 Ablation Study

In this subsection, we investigate the relative influences of different modules of ReSAG, we conduct ablation test for several variants as we mentioned before. The experimental results are also reported in the Table 1. It is noted that the performance for ReSAG (w/o SPN) decreases the most that denotes the importance of selection policy learning for the process of generation. In addition, we observe that the performance of ReSAG (w/o DPR-P) decreases a lot that indicates a good cold-start strategy for the retriever can significantly improve the system performance. Moreover, to evaluate the influence of different policy learning algorithms in the selection policy network, we use DQN to substitute for policy gradient algorithm. We find that DQN results in a decrease in performance.

5.3.3 Diversity Study

In our KQG, the diversity reflects the richness of knowledge among the generated queries. We report the evaluation of the diversity in the Table 2. It is observed that our ReSAG achieves the best performance on the Dist-2. But on Dist-1, ReSAG perform worse than T5. Overall, Our ReSAG achieve better performance compared with all baselines.

Table 2: Diversity evaluation results on datasets.

Approaches	Validation Dataset		Testing Dataset	
	Dist-1	Dist-2	Dist-1	Dist-2
Prompt4LLM	0.545	0.818	0.545	0.816
ICL	0.560	0.816	0.564	0.814
P-Tuning	0.342	0.532	0.321	0.562
LoraFtuneLLM	0.211	0.327	0.213	0.336
GPT2	0.280	0.408	0.281	0.411
T5	**0.696**	0.827	**0.698**	0.828
FID-KD	0.622	0.811	0.612	0.815
RetGen	0.628	0.820	0.619	0.818
ReSAG	0.636	**0.836**	0.634	**0.834**

However, it is challenging to evaluate the diversity of knowledge within the generated follow-up queries. To address this, we selected 100 samples at random from the testing dataset and enlisted the expertise of three human annotators. Their task was to determine the number of distinct knowledge points present in the generated follow-up queries. The findings revealed that our methods resulted in an average of 3.21 knowledge points per generated queries in comparison to the 4.67 knowledge points found in the ground-truth queries. This underscores the effectiveness of our approach in generating the follow-up queries with diverse knowledge.

5.4 Human Evaluation

We conduct user testing to assess the effectiveness of follow-up queries generated by our ReSAG for knowledge acquisition from a LLM. Specifically, we divided 8 real users into two groups, Group A and Group B. Each group was presented with 100 initial queries. In the first round, both Group A and Group B posed initial queries to an LLM. In the subsequent round, Group A had the option to either self-prompt for further knowledge or not, while Group B could choose to utilize the generated follow-up queries or not. Finally, we computed the average number of rounds. Experimental results reveal that Group B engaged in an average of 2.4 rounds of interactions for knowledge acquisition, while Group A only averaged 1.3 rounds. This demonstrates that the generated follow-up queries effectively enhance the interaction process between LLMs and users in progressive information acquisition.

5.5 Case Study

To demonstrate the effectiveness of our ReSAG, depicted in Figure 3, we conduct a case study on query generation using ChatGLM3 for knowledge acquisition. Initially, we input a initial query into ChatGLM3 to generate a response. Subsequently, we prompt ChatGLM3 to recommend five queries related to the initial query, serving as a baseline comparison known as Prompt4LLM. Upon observation, Prompt4LLM tends to suggest superficial queries of the "What", "How", and "Which" types, lacking depth in knowledge exploration. Conversely, our ReSAG effectively captures knowledge relations, generating insightful queries such as "Modified Gram-Schmidt QR" and "Projection of Linearly Independent Column Vectors onto the Range", which enrich and expand upon the initial query's understanding of "QR Decomposition". This case study underscores our ReSAG's ability to grasp knowledge relationships by retrieving and ranking relevant evidences. For further case studies, please refer to Appendix A.5.

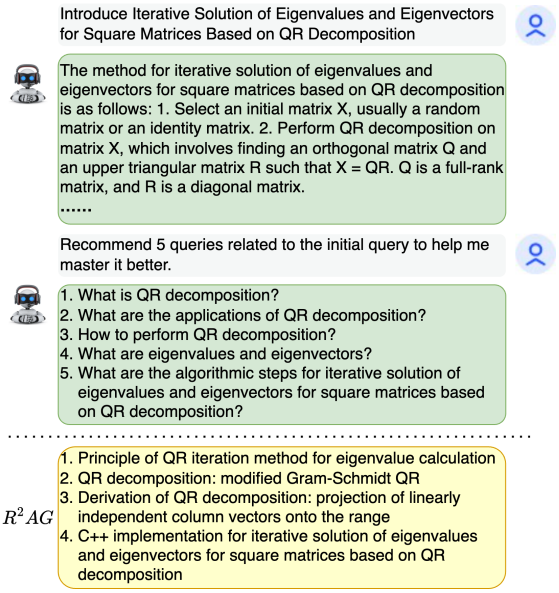

Figure 3: Case study of KQG for knowledge acquisition.

6 Conclusion

In this study, we introduced a novel approach named ReSAG, which integrates neural initial retrieval and selecting process into a T5-based sequence-to-sequence generation framework. ReSAG comprised a knowledge retriever, a selection policy network, and a query generator. Notably, the policy network served as a crucial bridge between the query generator and the knowledge retriever, ensuring their performance is correlated or aligned to enhance overall system performance. To improve the knowledge retriever, we leveraged signals from the policy network to guide progressive updates during training of the query generator. To enable end-to-end training, we introduced a novel variant of policy optimization that trains the initial retrieval and selector using only ground truth on the target sequence output. Extensive experiments demonstrated that our approach outperforms existing state-of-the-art methods, including generative models, RAG models, and LLM-based methods.

7 Limitations

While the effectiveness of the proposed approach has been validated within the domain of information technology, its applicability to other domains, such as general or educational question answering, remains to be explored. Additionally, although we employed both rule-based and machine learning methods to filter noise from the collected dataset to ensure high-quality training samples, some noise may still persist in the data.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Baichuan. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. *arXiv preprint arXiv:2207.06300*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Gautier Izacard and Edouard Grave. 2020a. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*.

Gautier Izacard and Edouard Grave. 2020b. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style,

high-performance deep learning library. *Advances in neural information processing systems*, 32.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Lin CY ROUGE. 2004. A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization of ACL, Spain*, volume 5.

Weizhou Shen, Yingqi Gao, Canbin Huang, Fanqi Wan, Xiaojun Quan, and Wei Bi. 2023. Retrieval-generation alignment for end-to-end task-oriented dialogue system. *arXiv preprint arXiv:2310.08877*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in neural information processing systems*, 27.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems*, 34:25968–25981.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hai Wan, Haicheng Chen, Jianfeng Du, Weilin Luo, and Rongzhen Ye. 2021. A dqn-based approach to finding precise evidences for fact verification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1030–1039.

Zezhong Wang, Fangkai Yang, Pu Zhao, Lu Wang, Jue Zhang, Mohit Garg, Qingwei Lin, and Dongmei Zhang. 2023. Empower large language model to perform better on industrial domain-specific question answering. *arXiv preprint arXiv:2305.11541*.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2022. Self-adaptive in-context learning. *arXiv preprint arXiv:2212.10375*.

Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu. 2018. Teaching machines to ask questions. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4546–4552. ijcai.org.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 247–256.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yizhe Zhang, Siqi Sun, Xiang Gao, Yuwei Fang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2022. Retgen: A joint framework for retrieval and grounded text generation modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11739–11747.

Zhuosheng Zhang, Hanqing Zhang, Keming Chen, Yuhang Guo, Jingyun Hua, Yulong Wang, and Ming Zhou. 2021. Mengzi: Towards lightweight yet ingenious pre-trained models for chinese. *arXiv preprint arXiv:2110.06696*.

Appendix

A.1 Dataset

A.1.1 Data Collection

The KQG task, unlike the RAG tasks, addresses the challenge of diverse evidence retrieval for multi-

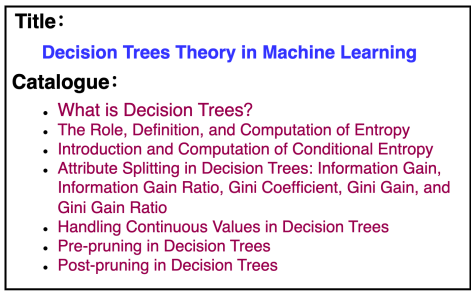

Figure 4: The case example of a title and its catalogue.

target generation. To facilitate this objective, we have curated a large-scale Chinese dataset specifically designed for follow-up query generation. The dataset is sourced from CSDN [3], an online knowledge sharing community focusing on information technology, where a substantial amount of knowledge articles is available. Each article comprises a title, a catalogue, and passages. Figure 4 illustrates an example of a title alongside its corresponding catalogue structure. Upon closer inspection, it becomes evident that the title functions akin to the initial query for a specific knowledge, and the catalogue serves as a set of follow-up queries with associated knowledge. Leveraging this observation, we extract the title as the initial query and consider each item in the catalogue as a target output. Consequently, we construct our dataset based on the extracted query and the list of target output. Furthermore, we create a document library to facilitate evidence retrieval, drawing on the text passages within the articles.

To ensure the high quality of samples, we employ two approaches to filter out low-quality samples: a heuristic rules-based method and a LLM-based in-context learning method [4]. The former utilizes regular expressions or empirical rules to filter titles or catalogues containing specific symbols, or sentences that are excessively long or short. The latter seeks to leverage the semantic understanding of a LLM to assess the quality of a sample, incorporating a few manually annotated high-quality demonstrations for guidance.

A.1.2 Statistics and Analysis

In this subsection, we conduct an analysis of the statistical features of our constructed dataset. After quality filtering, our KQG dataset comprises a total of 210,531 (query, follow-up queries) pairs and 1,677,513 document passages. The samples were randomly split, resulting in 189,477, 10,577, and 10,577 samples allocated to the training, development, and test sets, respectively. The details, including queries, follow-up queries, text passages, and article URLs, will be provided for research purposes as we make the dataset public.

Table 3: The statistics of the KQG dataset.

Data	#Num	#Len(Query/FQ)	#FQ
Training Set	189,477	17.76/ 14.44	9.51
Validation Set	10,527	17.70/14.35	9.46
Testing Set	10,527	17.83/14.50	9.51
Passages	1,677,513	580.67	-

The key statistical information is presented in Table 3. Notably, the average length of textual passages in the document library is 580.67. For the training, validation, and testing sets, the average length of queries and target output are as follows: 17.76/14.44, 17.70/14.35, and 17.83/14.50, respectively. On average, each query corresponds to 9.51, 9.46, and 9.51 follow-up queries for the training, validation, and testing sets, respectively. To ensure knowledge diversity, the dataset encompasses various topics in information technology, covering areas such as front-end development, back-end development, algorithms, databases, operating systems, software development, and big data. The constructed dataset is approximately 2.3 GB. This high-quality dataset encompasses a diverse range of knowledge topics, making it adequately representative for our KQG task.

A.2 RAG

RAG models use the input query x to retrieve text documents z from document library Z and use them as additional context when generating the target output y. Formally, the optimization process is carried out based on the following formulation:

$$P(y|x;\eta,\theta) = \sum_{z\in Z} P_\theta(y|x,z)P_\eta(z|x), \qquad (17)$$

where $P_\eta(z|x)$ is the retriever with parameters η and $P_\theta(y|x,z)$ denotes the generator parametrized by θ that generates the target y based on the original input query x and the retrieved document z. In practical scenarios, Z often encompasses a large volume of documents, making exhaustive enumeration over Z impractical. Instead, a dense document retriever $P_\eta(\cdot)$ is employed to significantly narrow down the search to a select few relevant documents. Specifically, the retriever P_η takes Z and x as input, producing relevance scores $\{s_1, ..., s_K\}$ for

[3] https://www.csdn.net/

[4] We utilize ChatGLM3 for Chinese comprehension.

the top-K (K being a hyperparameter) documents $\tilde{Z} = \{z_1, \cdots, z_K\}$.

Given x and the top-K retrieved documents $\{z_1$, z_2, ..., $z_K\}$, the generator $P_\theta(\cdot)$ is used to produce a probability score for a given reference target y, i.e., $P_\theta(y|x, z_i)$. The loss can be approximated as:

$$\mathcal{L}(\eta, \theta) = -log \sum_{i=1}^{K} P_\theta(y|x, z_i) P_\eta(z_i|x), \qquad (18)$$

where the normalized probability $p(z_j|x)$ is defined as $exp(s_j)/\sum_{i=1}^{K} exp(s_i)$, with relevance scores s serving as logits. The set $\widetilde{Z} = \{z_1, \cdots, z_K\}$ is obtained from the retrieval process using $P_\eta(Z, x)$.

Prior RAG models (Lewis et al., 2020; Singh et al., 2021; Zhang et al., 2022) focus on co-training the retriever and generator in an end-to-end differentiable manner, incorporating feedback from the model itself as "pseudo labels" to optimize retriever and reader parameters iteratively.

A.3 Baseline Description

We provide a brief description of the baselines used in our experiments:

- **GPT2** (Radford et al., 2019) is a pretrained language model based on the Transformer decoder, which has achieved outstanding performance in a wide range of text generation tasks.
- **T5** (Raffel et al., 2020) is implemented by typical encoder-decoder Transformer, solving text-to-text tasks. We initialize the parameters of T5 with a pretrained Chinese T5 base model, i.e., a mengzi-t5-base (Zhang et al., 2021).
- **P-Tuning (Liu et al., 2022)** is a method that tunes continuous prompts with a frozen language model for query generation. In our case, we have opted for ChatGLM3 (Du et al., 2021) as our base model.
- **LoraFtuneLLM** is a method that utilizes LoRA (Hu et al., 2021) technique to finetune a large language model to generate the follow-up queries.
- **Prompt4LLM** is a method that directly prompts a large language model to generate the follow-up queries related to the initial query.
- **ICL (Wu et al., 2022)** is a method that utilizes in-context learning technique to prompt a large language model to generate the follow-up queries with a few demonstration examples sampling from the training set.
- **FID-KD (Izacard and Grave, 2020a)** is retrieval-augmented generation approach which learns retriever models for generation tasks by knowledge distillation technique without requiring annotated pairs of query and passages.
- **RetGen (Zhang et al., 2022)** is retrieval-augmented generation which adopts a joint training framework for text generation and passage retrieval utilizing a language model signal.

A.4 Evaluation Metric Description

We provide a brief summary of the metrics used in our experiments:

- **BLEU-4** (Papineni et al., 2002): This metric utilizes n-gram precision to evaluate the similarity between a generated text and a reference text. It counts the occurrences of unigrams, bigrams, trigrams, and four-grams that match their corresponding counterparts in the reference text.
- **ROUGE (ROUGE, 2004)**: This metric includes ROUGE-1, ROUGE-2 and ROUGE-L. ROUGE-1 and ROUGE-2 measure overlap of unigrams and bigrams respectively. ROUGE-L measures overlap of the longest common subsequence between a generated text and a reference text.
- **BERTScore** (Zhang et al., 2019): Utilizing contextual embeddings to compute token similarity, BERTScore is reported based on bert-base-chinese [5] in our scenario.
- **Diversity** (Li et al., 2015): We report degree of diversity by calculating the number of distinct unigrams and bigrams in generated responses. The value is scaled by total number of generated tokens to avoid favoring long sentences (shown as Dist-1 and Dist-2 in Table 2.).

A.5 More Case Studies

We conduct additional case studies, as depicted in Figure 5, to further validate the efficacy of our ReSAG. In both cases in Figure 5, the ChatGLM3-based prompting approach struggle to capture the depth of knowledge inherent in the initial query. In contrast, our ReSAG adeptly generated diverse follow-up queries that encapsulated more comprehensive knowledge. For instance, in the top case of Figure 5, crucial aspects such as "Selection of rated current," "Overcurrent protection," and "Rated voltage" are pivotal considerations for IGBT selection.

[5] https://huggingface.co/bert-base-chinese

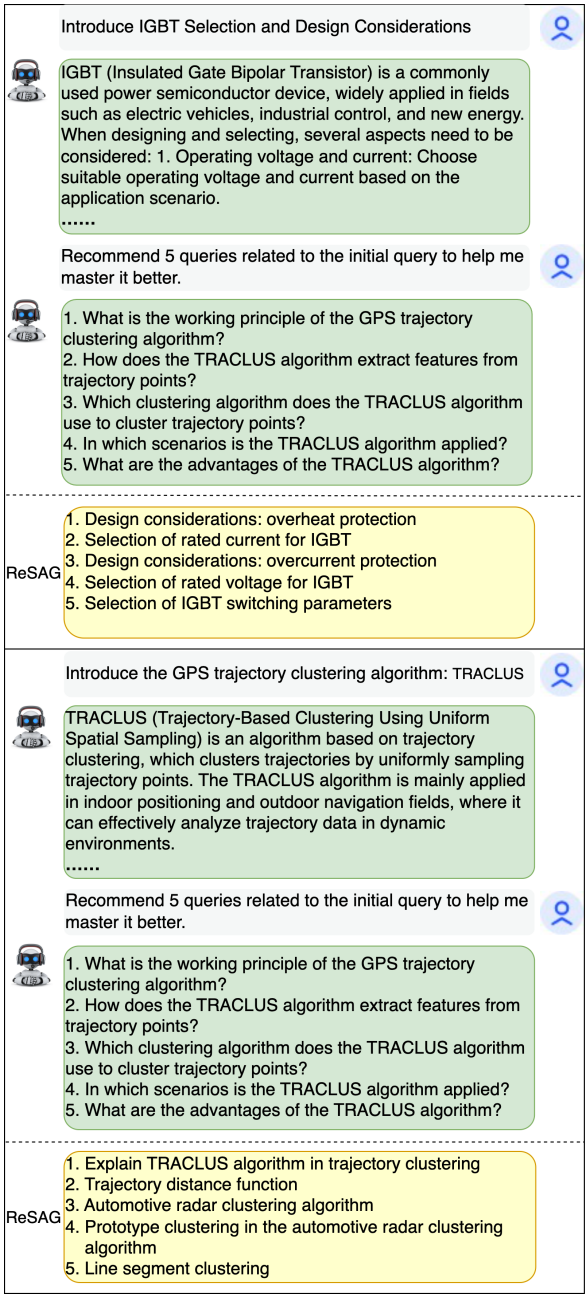

Figure 5: The additional case studies of query generation based on ChatGLM3.

A.6 Implementation Details

A.6.1 Hardware and library

We conducted all experiments on a machine equipped with 56 CPUs, 1.0TB of physical memory, and 4 A100 GPUs. Our ReSAG and baseline models were implemented using PyTorch (Paszke et al., 2019) [6].

A.6.2 Model configurations

We employ the base configuration for both the knowledge retriever and query generator, comprising 12 layers, 768-dimensional hidden size, and 12 attention heads. As for the selecting policy network, the context module utilizes a 3-layer Transformer encoder with 12 heads and a 768-dimensional hidden size. The LSTM hidden size is also set to 768. In our experiments, we initially retrieve the top 50 passage as evidence and select 5 passages among them, i.e., $N = 50$ and $K = 5$. Due to GPU memory constraints, we limit our experiments to the base configuration. Nonetheless, we anticipate that our findings would extend to larger configurations.

Algorithm 1 The Algorithm Training Procedure for ReSAG

Input: The passage knowledge base $\mathcal{Z}$, $(X, Y) \in \mathcal{D}$, a LLM F_{lm}, the retriever P_η, the selector P_ϕ, and the generator P_θ
Output: The trained parameters η, ϕ and θ

Initialize η and θ with a Chinese-Roberta-wwm-ext and a mengzi-t5-base, respectively
Pre-train P_η with contrastive learning in the SimCSE (Gao et al., 2021)
Pre-train P_θ only using the initial query as input and follow-up queries as targets
Compute the passage index using the current retriever parameters η
for all $j \in |\mathcal{D}|$ and $(X, Y) \in \mathcal{D}$ **do**
    Retrieve top-N passages $\tilde{\mathcal{Z}}$ using the current retriever parameters η
    Compute top-K passages $\hat{\mathcal{Z}}$ using the current selector parameters ϕ
    Generate follow-up queries $\hat{Y}$ based on $\hat{\mathcal{Z}}$ and X
    Compute the reward between Y and $\hat{Y}$ using Eq. 16
    Optimize the parameter θ using maximum likelihood estimation between Y and $\hat{Y}$ using Eq. 8
    Optimize the parameters η and ϕ using Eq. 15
    if $j\%1000 == 0$ **then**
        Update the passage index using the updated retriever parameters η
    end if
end for

A.6.3 Retrieval

To facilitate rapid retrieval, we pre-computed embeddings for 1,677,513 evidence passages, forming our passage index. Through experimentation, we discovered that conducting Maximum Inner Product Search with FAISS is sufficiently efficient on CPU. Consequently, we store the passage index vectors on the CPU, necessitating 16 GB of CPU memory for our dataset.

A.6.4 Training Details

We initially set the retriever parameters using a pre-trained Chinese BERT model [7], followed by pre-training with contrastive learning techniques

[6] The dataset and source codes will be made available upon acceptance of our work.

[7] https://huggingface.co/hfl/chinese-roberta-wwm-ext

in SimCSE (Gao et al., 2021). For the query generator, we initialize the T5 parameters with a pre-trained Chinese T5 base model (Zhang et al., 2021), i.e., the mengzi-t5-base [8]. Subsequently, we exclusively pre-train it using the initial query as input and follow-up queries as targets. During end-to-end training, as the passage encoder parameters are continuously updated, pre-computed passage embeddings may become outdated. To address this, we asynchronously compute fresh passage embeddings using the latest passage encoder checkpoint, updating the passage index every 1000 training steps to maintain relevance.

A.6.5 Inference

At inference time, we employ greedy decoding for follow-up query generation.

A.7 Algorithm Training Procedure

The detailed training procedure is shown in Algorithm 1.

[8] https://huggingface.co/Langboat/mengzi-t5-base