HyQE: Ranking Contexts with Hypothetical Query Embeddings

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) systems can effectively address user queries by leveraging indexed document corpora to retrieve the relevant contexts. Ranking techniques have been adopted in RAG systems to sort the retrieved contexts by their relevance to the query so that users can select the most useful contexts for their downstream tasks. While many existing ranking methods rely on the similarity between the embedding vectors of the context and query to measure relevance, it is important to note that similarity does not equate to relevance in all scenarios. Some ranking methods use large language models (LLMs) to rank the contexts by putting the query and the candidate contexts in the prompt and asking LLM about their relevance. The scalability of those methods is contingent on the number of candidate contexts and the context window of those LLMs. Also, those methods require fine-tuning the LLMs, which can be computationally expensive and require domain-related data. In this work, we propose a scalable ranking framework that does not involve LLM training. Our framework uses an off-the-shelf LLM to hypothesize the user's query based on the retrieved contexts and ranks the contexts based on the similarity between the hypothesized queries and the user query. Our framework is efficient at inference time and is compatible with many other context retrieval and ranking techniques. Experimental results show that our method improves the ranking performance of retrieval systems in multiple benchmarks.

1 Introduction

RAG systems have gained significant attention in natural language processing (NLP) research. It empowers LLMs to draw information from a wider spectrum of knowledge beyond the context windows of LLMs. Standard RAG systems utilize context retrieval techniques to efficiently fetch from dedicated databases the contexts that are relevant to user-given queries. Then, LLMs can generate responses to those queries based on the retrieved contexts, reducing the likelihood of generating hallucinated information (Ram et al., 2023; Asai et al., 2023a). This integration of retrieval and generation components in RAG systems allows them to demonstrate superior performance in tasks that may require retrieving information from prohibitively long contexts.

The accuracy of ranking the relevance of the context is a core determinant of the performance of RAG systems (Shi et al., 2023). Classical retrieval methods such as TF-IDF and BM25 (Robertson and Zaragoza, 2009) rely on lexical similarities to rank contexts. Recent advancements in embedding models such as BERT (Kenton and Toutanova, 2019; Reimers and Gurevych, 2019) have enabled the capture of the semantic similarity between texts through dense vector representations. To improve the zero-shot performance in unseen contexts, Contriever (Izacard et al., 2021) and other successive embedding models are trained via contrastive learning techniques. However, retrieval with these embedding models focuses on similarity, but similarity alone does not always ensure that the context effectively addresses the query.

LLMs have been incorporated to address this issue. For instance, LLM-based re-rankers (Sun et al., 2023; Pradeep et al., 2023) can determine whether a context addresses a query better than others. However, those re-rankers require fine-tuning, which demands extensive dedicated datasets and significant computational resources. Other methods include using LLM to expand the query before retrieval. HyDE (Gao et al., 2023a), for instance, utilizes an LLM to generate hypothetical contexts based on the query, subsequently retrieving concrete contexts that are close to these hypothetical contexts in the embedding space. However, the LLM must have sufficient background knowledge about the context to be retrieved so that it can generate semantically similar contexts. Otherwise, the hypothesis space of the generated contexts would be indefinitely large, and the LLM can generate outdated, irrelevant, hallucinated, and even counterfactual contexts (Brown et al., 2020; Mallen et al., 2022). We provide an example later in Fig.3(b), where GPT-3.5-turbo generates outdated information that fails to reflect recent developments on a specific topic.

In this paper, we propose a novel context ranking framework. Our approach uses an LLM to generate hypothetical queries based on the existing contexts. It then measures the relevance between the context and any user-given query based on the similarity between the hypothetical queries and the user-given query. While our method does not require the LLM to have prior knowledge about the query or the context, the hallucination of the LLM is restrained since a context has limited information and can only provide answers to a certain range of queries. Furthermore, while HyDE has to use an LLM to generate hypothetical contexts online for every input query, our approach allows retrieving previously generated hypothetical queries for future input queries. Our method also differs from the LLM-based re-ranker in two-fold. First, our method does not require fine-tuning an LLM. Second, our approach uses text embedding for ranking, while an LLM-based re-ranker has to call an LLM to answer the relevance between every input query and context. However, our method can be used in conjunction with other ranking methods to iteratively refine the ranking of the retrieved contexts.

Besides introducing our approach, we compare our approach with existing approaches from the theoretical lens. We analyze the causality relationship between the queries and contexts within a class of context ranking approaches, identifying their potential issues, such as their susceptibility to spurious causality relationships. We then show that our approach mitigates these issues by following a variational inference approach. Our experimental results demonstrate improvements in ranking the retrieved contexts across multiple information retrieval benchmarks while maintaining efficiency and scalability. Our major contribution is listed as follows.

- We propose to use LLMs to generate hypothetical queries and rank contexts by comparing the similarity between input queries and hypothetical queries.
- We examine the causal relationships between queries and contexts in existing context ranking methods and develop a variational inference framework for context ranking.
- We evaluate our method in multiple information retrieval benchmarks by combining different embedding models with different LLMs. The results show that our method can improve the ranking accuracy in most of the benchmarks.

2 Related Work

Retrieval Augmented Generation (RAG) systems have become a focal point in NLP research, enhancing LLMs by accessing broader knowledge bases beyond LLM context windows (Lewis et al., 2020; Gao et al., 2023b). These systems use information retrieval techniques to fetch relevant contexts from dedicated databases based on user queries, improving performance in tasks requiring extensive context (Mialon et al., 2023).

Information Retrieval methods, such as TF-IDF and BM25, rely on lexical similarities to rank contexts (Robertson and Zaragoza, 2009). Recent advancements in embedding models such as BERT (Kenton and Toutanova, 2019; Reimers and Gurevych, 2019) allow capturing text semantics through dense vector representations (Asai et al., 2021). Contrastive learning techniques have further improved the zero-shot performance of embedding models such as Contriever (Izacard et al., 2021) in unseen contexts by training the models to differentiate between similar and dissimilar contexts (Gao et al., 2021).

Document Expansion and Query Expansion are classical techniques to improve retrieval quality and have been widely adopted in RAG systems (Wang et al., 2023). Query expansion, which dates back to (Carpineto and Romano, 2012), typically involves rewriting the query based on labels (Lavrenko and Croft, 2001). When labels are not available, the query can be expanded with generated contexts (Liu et al., 2022). For instance, HyDE (Gao et al., 2023a) uses LLMs to generate hypothetical contexts based on the input query and uses the embeddings of the query and the hypothetical contexts for retrieval. However, when the LLM lacks knowledge about the query, query expansion can be susceptible to hallucinated or counterfactual content (Brown et al., 2020).

Document expansion (Nogueira et al., 2019) involves appending each context with a generated query and creating indexes for the expanded context in the database. Our framework also generates queries based on the contexts but does not expand the contexts. Studies on generating high-quality queries to build synthetic datasets (Almeida and Matos, 2024) can be helpful for our framework, but that is not the focus of this paper.

Large Language Models (LLMs), from the small-size open source models such as Mistral-7b (Jiang et al., 2023) to the large-size proprietary models such as GPT-4 (Achiam et al., 2023), are pre-trained on trillions of tokens, exhibiting unparalleled emergent and generalization abilities across tasks (Schaeffer et al., 2023). LLMs can be fine-tuned to rank the relevancy between contexts and queries (Asai et al., 2023b; Sun et al., 2023; Pradeep et al., 2023). Although effective, fine-tuning requires significant computational resources and extensive annotated data (Bajaj et al., 2016). Furthermore, those methods have to face the challenges related to the context window size (Wang et al., 2024; Kaddour et al., 2023; Child et al., 2019; Gu and Dao, 2023), as they combine the query and contexts into a single prompt. Our method does not use LLMs to evaluate the query-context relevancy.

Variational Inference (Blei et al., 2017) sits at the core of our proposed framework. It has been extensively studied across many fields of machine learning (Kingma and Welling, 2013; Hoffman et al., 2013; Zhou and Li, 2022; Fellows et al., 2019). In this work, we treat queries and contexts as random variables with causal relationships and reformulate the ranking problem as a probability inference problem. It is widely known that generative models that respect the causality relationships are more robust to distribution shifts because they can avoid learning spurious relationships between random variables (Ahuja et al., 2021; Schölkopf et al., 2021; Lu et al., 2022). In this work, we use an LLM to simulate the query-context relationship while avoiding the intervention of prior knowledge, thereby preserving the causal structure.

3 Background

A RAG system retrieves information from a document corpus $C = \{c_1, c_2, \ldots, c_i, \ldots\}$ where each c_i is a context. Assuming that Q is the whole set of user input queries, given an input query $q \in Q$, a retriever returns a ranked list of relevant contexts from C. The ranking of those contexts can be evaluated by using Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2002) which measures the ranking with graded relevance. After the retrieval step, one or multiple ranking procedures can be adopted to iteratively refine the quality of the ranking. We assume that each ranking procedure, including that during retrieval, uses a scoring function $r_q : C \to \mathbb{R}$ to quantify the relevance between any context $c \in C$ and the query q. We can rank the contexts with this r_q, i.e., $\forall c_1, c_2 \in C, c_1 \preceq c_2 \Leftrightarrow r(q, c_1) \leq r_q(c_2)$.

A ranker can only target the first K contexts as ordered by some scoring function of the previous ranking procedure. We denote the set that includes the first K contexts as $C_{q,K} \subseteq C$ such that $|C_{q,K}| = K$. After ranking those contexts, a new scoring function r_q over $C_{q,K}$ is generated.

When using an embedding model for ranking, we use the cosine similarity between query embedding and context embedding to determine the relevance of the query and context. We use E to denote the embedding model. The cosine similarity between a query q and a context c is

$$sim(q, c) = \frac{\langle E(q), E(c) \rangle}{||E(q)||_2 \cdot ||E(c)||_2} \quad (1)$$

As a result, given any query q, an embedding model-based ranker's scoring function is defined as $r_q(c) = sim(q, c)$.

4 Method

In this section, we introduce our framework for ranking contexts with hypothetical queries. We first illustrate our context ranking procedure, explain how to obtain those hypothetical queries, and then discuss its complexity.

For each context $c \in C$, we hypothesize the probable queries that the context c can address or the topics it discusses. We refer to these queries as hypothetical queries, denoted as $\hat{q}$. For each $c \in C$, we let $H(c)$ denote the set of hypothetical queries associated with c. Our ranking method determines the relevance of a given query q and a context c based on the similarity between the embedding of q and the embedding of c, as well as the similarity between those of q and the hypothetical queries $H(c)$ as in Eq.2 where we introduce a hyperparameter λ to balance the two similarities.

$$r_q(c) := sim(q, c) + \lambda \cdot \max_{\hat{q} \in H(c)} sim(\hat{q}, q) \quad (2)$$

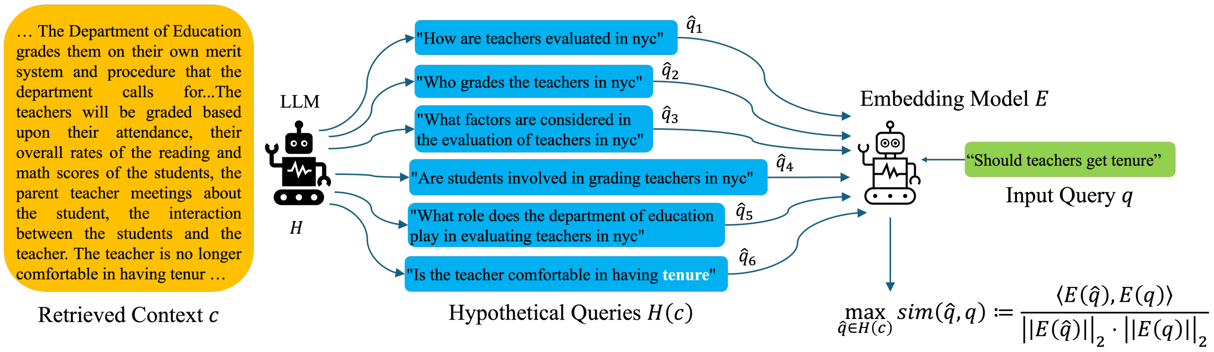


Figure 1: A flow chart of HyQE ranking framework. Given a query q and a retrieved context c, an LLM H is used to generate a set of hypothetical queries $\hat{q}$ from c. Then an embedding model E is used to evaluate the semantic similarity between q and $\hat{q}$'s. Then cosine similarity is used to determine whether c is relevant to q as in Eq.2.

Algorithm 1 outlines our context ranking procedure. We start with a set $C_{q,K}$ of K candidate contexts, which are typically the top-K results from a prior ranking step. For each context $c \in C_{q,K}$, we generate a set of hypothetical queries $H(c)$ by using an LLM, compute the embedding of c and each $\hat{q} \in H(c)$ with an embedding model E, and then calculate the relevance score $r_q(c)$ using Eq.1.

Algorithm 1 HyQE

1: **Input:** A query q; a set $C_{q,K}$ of K candidate contexts; an LLM H; an embedding model E
2: **foreach** context c in $C_{q,K}$
3: Compute $sim(q, c)$ via Eq.1
4: Collect hypothetical queries $H(c)$
5: Compute $r_q(c)$ via Eq.2
6: Order $C_{q,K}$ by $r_q(c)$
7: **Output:** The ordered-set $C_{q,K}$

Hypothetical Query Generation. Our framework allows utilizing various LLMs to generate hypothetical queries ranging from Mistral 7b to GPT-3.5 and GPT-4. Fig.1(a) shows a flowchart of this query generation process. For each context c, we generate a set of hypothetical queries $H(c)$ by instructing an LLM H. Specifically, we use a single prompt to generate multiple queries for each context to avoid generating repetitive queries, as shown in Fig.2. The prompt is designed to ensure that the generated queries are diverse and relevant to the given context. If the length of the context c and the lengths of queries to be generated will exceed the window size of the LLM, we partition c, call the LLM to generate queries for one portion at a time, and collect all generated queries in the end.

Complexity. Although generating hypothetical queries for each c with an LLM can be time-

```
Which kinds of questions can be answered
based on the following passage

```<passage>
{context}
</passage>'''

Questions must be very short, different,
and be written on separate lines.
If the passage provides no meaningful
content, respond with a 'No Content'.
```

Figure 2: Prompt for hypothetical query generation. '{context}' is the placeholder for the context to be filled.

consuming, this overhead can be mitigated. Since the hypothetical queries $H(c)$ are independent of the input $q$, once $H(c)$ and the corresponding embeddings are obtained, they can be stored and reused for future queries that involve the same context $c$. This eliminates the repetitive query-generation step in line 4 of the algorithm. When a previously seen context $c$ is retrieved for some new input query $q'$, we can quickly retrieve the stored $H(c)$ and embeddings of the queries in $H(c)$. Then we only need to perform a similarity search to find the hypothetical query $h \in H(c)$ with the closest embedding to the new query $q'$ to compute $r_{q'}(c)$ in line 5. By leveraging stored hypothetical queries and their embeddings, our framework ensures efficient and scalable query processing, reducing the computational overhead of real-time LLM calls.

The complexity of our ranking framework can be broken down as follows. Generating hypothetical queries $H(c)$ for each context $c \in C$ and computing their embeddings can incur a one-time computational cost. If each context $c$ can generate $M$ hypothetical queries, the total complexity of this one-time computation is $O(|C| \cdot M)$ where $|C|$ is
```

the total number of contexts and M is limited by the information encompassed in the context. This complexity is amortized as the number of input queries increases, making our approach more scalable. If a c is retrieved for a new query q' and its hypothetical query set $H(c)$ has been indexed, the one-time computational complexity of retrieving the closest hypothetical query $\hat{q} \in H(c)$ via similarity search is typically sub-linear in $|H(c)|$.

In comparison, query expansion methods typically generate contexts for each input query. Thus, the total complexity cannot be amortized by the growing number of input queries. For instance, HyDE (Gao et al., 2023a) requires generating a group of hypothetical documents for each query, leading to a total complexity of $O(|Q| \cdot N)$ where $|Q|$ is the number of queries and N is the number of hypothetical contexts, both of which are independent of the document corpus C and can be infinite.

Similar comparisons apply to LLM-based re-rankers (Sun et al., 2023; Pradeep et al., 2023), where the complexity is proportional to the lengths of the input query and the retrieved contexts, as the re-rankers require concatenating the query and contexts in the prompts to generate responses. This complexity cannot be amortized, making ranking contexts expensive as the number of queries increases, considering that LLM-based re-rankers are often large, proprietary models. HyQE allows using small pre-trained LLMs and open-source embedding models, significantly reducing operational costs while maintaining efficiency and effectiveness.

5 A Variational Inference Perspective

In this section, we explain how to use variational inference to derive Eq.2 by establishing the causal relationship between queries and contexts.

5.1 Causal Relationship between Queries and Contexts

In Fig.3, we treat context c and query q as two random variables. We can think of calculating the ranking score $r_q(c)$ as measuring the probability $p(c|q)$ of context c answering question q in the causality model. Different ranking methods model the causal relationship between c and q in different ways, resulting in different $p(c|q)$ and $r_q(c)$. For instance, the standalone cosine similarity $sim(q, c)$ can produce a spurious $p(c|q)$ since c and q being similar does not necessarily imply that c provides answers to q, as shown by the example in Fig.3(a). Query expansion methods such as HyDE (Gao et al., 2023a) introduce a hypothetical context $\hat{c}$ as a latent variable and employ a generative model to simulate $p(\hat{c}|q)$. However, this causality modeling inevitably involves LLM's prior knowledge as an intervention (Wachter et al., 2017), which can lead to spurious causality. The external knowledge from LLM is represented as an additional variable D from another context space that is indefinitely larger than that of c. As shown in Fig.3(b), it can influence the generation of $\hat{c}$ by introducing outdated, irrelevant, or even counterfactual information (Brown et al., 2020).

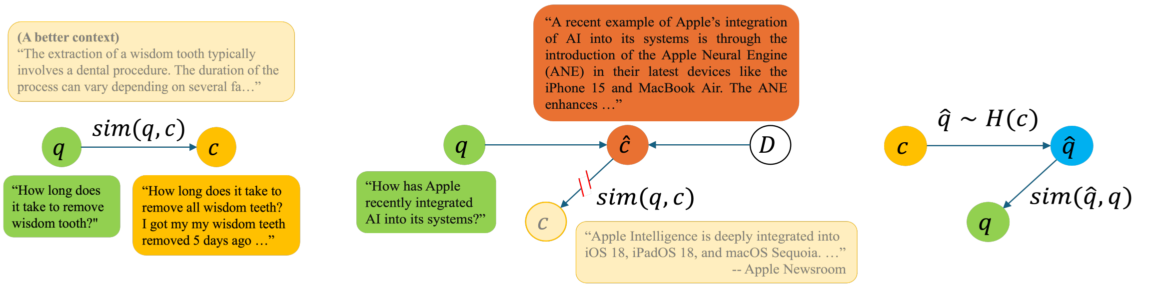
In contrast, HyQE, as shown in Fig.3(c), introduces a hypothetical query $\hat{q}$ as a latent variable and employs a generative model to simulate $p(\hat{q}|c)$ without involving the prior knowledge of the LLM. This confines the generation of hypothetical query $\hat{q}$ strictly within the scope of the context c, avoiding the pitfalls of spurious causality and ensuring that the causal relationships remain accurate and relevant. This allows us to use cosine similarity to simulate $p(q|\hat{q})$ where $\hat{q}$ and q are both queries.

5.2 Ranking Contexts from a Variational Inference Perspective

Now we show how we derive Eq.2 based on Fig.3(c). Given a user query q and a context set $C_{q,K}$, we define $p(c)$ as some prior confidence over the context set $C_{q,K}$ that satisfies $p(c) \propto \exp(sim(q, c))$. We let $p(q|c)$ be the probability of context c providing answers to the query q, and let $p(q)$ be some prior probability of accepting an input query q, which can be seen as a constant when q is already given. Based on $p(c)$, $p(q|c)$, and $p(q)$, we aim to learn $p(c|q) := p(c)p(q|c)/p(q)$, which can be seen as the confidence of the context c addressing the given query q. Then, we learn $p(c|q)$ by finding a distribution $p_q(c)$ that matches $p(c|q)$ so that we can establish a scoring function based on $p_q(c)$, i.e., $r_q(c) \propto \log p_q(c)$. This learning objective can be formulated as minimizing the KL-divergence $D_{KL}(p_q(c)||p(c|q))$ which can be achieved by maximizing the evidence lower-bound (ELBO) of $D_{KL}(p_q(c)||p(c|q))$ as shown in Eq.3.

$$ELBO(r_q) := D_{KL}(p_q(c)||p(c)) - \mathbb{E}_{c\sim p_q(c)}[\log p(q|c)] \quad (3)$$

Eq.3 uses a regularization term $D_{KL}(p_q(c)||p(c))$ to penalize p_q if $p_q(c)$ deviates from $p(c)$. Therefore, we include $sim(q, c)$ as a part of r_q such that



(a) Cosine Similarity (b) Query Expansion (c) HyQE

Figure 3: The random variables c and q respectively indicate context and user input query. (a) Cosine similarity prioritizes semantic similarity rather than retrieving a better context for answering the query. (b) The causality relationship in query expansion methods such as HyDE. The random variable $\hat{c}$ is a hypothetical context, and D indicates the prior knowledge of the LLM used to generate $\hat{c}$. In this example, we use GPT-3.5-turbo to generate a hypothetical context $\hat{c}$ to answer the question in q. However, $\hat{c}$ contains outdated information and cannot be used to retrieve the most relevant context c through semantic search. (c) The causality relationship in HyQE. An LLM H is used to generate the hypothetical query $\hat{q}$. The causal relationship q and $\hat{q}$ can be simulated with causal similarity.

the greater $p(c) \propto \exp(sim(q, c))$ is, the greater $p_q(c) \propto \exp(r_q(c))$ becomes. Meanwhile, the second term in Eq.3 indicates that p_q should also align with $p(q|c)$, the probability of c providing answers to q. To estimate $p(q|c)$, we factorize $\log p(q|c) = \log \mathbb{E}_{\hat{q} \sim p(\hat{q}|c)}[p(q|\hat{q})]$ where $p(\hat{q}|c)$ is the probability of c addressing a hypothetical query $\hat{q}$ and $p(q|\hat{q})$ is the probability of obtaining an input query q given that the semantics of q is equivalent to a given hypothetical query $\hat{q}$. We can safely use semantic similarity to approximate relevance between queries, i.e., $p(q|\hat{q}) \propto \exp(sim(\hat{q}, q))$. We estimate the expectation w.r.t $p(\hat{q}|c)$ by uniformly sampling from the set $H(c)$ of hypothetical queries such that $\log p(q|c) = \log \mathbb{E}_{h \sim p(\hat{q}|c)}[p(q|\hat{q})] \approx \log \frac{1}{|H(c)|} \sum_{\hat{q} \in H(c)} p(q|\hat{q})$. We then have the following two options for further approximation:

Option 1. Based on the soft-max approximation, $\log \frac{1}{|H(c)|} \sum_{\hat{q} \in H(c)} p(q|\hat{q}) \approx \max_{\hat{q} \in H(c)} \log p(q|\hat{q}) = \lambda \cdot \max_{\hat{q} \in H(c)} sim(h, q) + const$ where λ is a hyperprameter. Then we recover Eq.2 by ignoring the constant and adding $sim(\hat{q}, q)$ mentioned earlier.

Option 2. Based on Jensen's inequality (Jensen, 1906), we derive a lower bound of the estimated $\log p(q|c)$ as shown in Eq 4, This allows us to maximize ELBO in Eq.3 by maximizing Eq.4, resulting in an alternative of Eq.2 as shown in Eq.5.

$$\begin{aligned} &\log \tfrac{1}{|H(c)|} \textstyle\sum_{\hat{q} \in H(c)} p(q|\hat{q}) \\ &\geq \tfrac{1}{|H(c)|} \textstyle\sum_{\hat{q} \in H(c)} \log p(q|\hat{q}) \\ &= \lambda \cdot \tfrac{1}{|H(c)|} \textstyle\sum_{\hat{q} \in H(c)} sim(q, \hat{q}) + const \end{aligned} \quad (4)$$

In our HyQE framework, we mainly focus on Option 1. We will compare Option 1 with Option 2 in our evaluation.

$$\begin{aligned} r_q(c) := &\ sim(q, c) + \\ &\lambda \cdot \frac{1}{|H(c)|} \sum_{\hat{q} \in H(c)} sim(q, \hat{q}) \end{aligned} \quad (5)$$

6 Experiments

We test our method on multiple benchmarks to investigate the main question: *whether HyQE improves the nDCG@10 in the benchmarks?* In addition, we also investigate the following questions.

A. Does changing the LLMs influence the results?
B. Does changing the λ in Eq.2 influence the results?
C. Is HyQE compatible with different retrieval methods such as HyDE (Gao et al., 2023a)?
D. How well does Eq.5 perform in comparison with Eq.2?

Datasets. We test our methods on the following datasets: COVID (Thakur et al., 2021), NEWS (Thakur et al., 2021), Touche2020 (Thakur et al., 2021), DL19 (Craswell et al., 2020), and DL20 (Craswell et al., 2020). We use the same prompt for all the datasets except for the touche2020 dataset, in which the queries represent topics of arguments while the contexts consist of dialogues in those arguments. The prompt designed for this dataset can be found in Appendix B.

Baselines. We use two kinds of retrievers: one is embedding model-based retrievers, including contriever and bge-base-en-v1.5; the other is SPLADE++_EnsembleDistil (Formal et al., 2022),

Retrieval Model	Embedding Model	HyQE Model	DL19	DL20	COVID	NEWS	Touche
contriever	contriever	-	44.54	42.13	27.32	34.84	16.68
		GPT-4o	53.97	51.93	35.03	41.27	17.78
		GPT-3.5-turbo	53.19	50.04	35.06	42.33	21.02
		Mistral-7b-instruct	52.28	49.62	35.54	42.56	20.78
bge-base-en-v1.5	bge-base-en-v1.5	-	70.39	68.30	69.96	40.94	18.99
		GPT-4o	72.04	69.42	80.29	43.01	19.44
		GPT-3.5-turbo	71.77	68.33	80.13	44.03	20.14
		Mistral-7b-instruct	70.72	69.02	78.93	43.34	21.36
SPLADE++ ED	contriever	-	53.47	53.51	67.35	39.01	20.45
		GPT-4o	60.68	61.66	64.90	44.45	19.17
		GPT-3.5-turbo	60.08	58.27	65.97	44.79	23.01
		Mistral-7b-instruct	57.99	59.59	65.78	44.33	22.32
	bge-base-en-v1.5	-	71.25	68.58	80.45	46.21	21.53
		GPT-4o	72.35	68.96	80.82	46.25	22.11
		GPT-3.5-turbo	71.66	68.83	81.55	46.18	23.15
		Mistral-7b-instruct	71.78	69.06	80.82	45.97	22.80
	E5-large-v2	-	70.18	72.50	76.73	40.65	18.03
		GPT-4o	72.69	71.46	75.87	50.43	20.50
		GPT-3.5-turbo	72.23	71.88	78.29	50.16	23.08
		Mistral-7b-instruct	69.92	72.97	76.90	48.67	22.52
	nomic-embed-text-v1.5	-	66.68	67.28	79.37	45.80	23.93
		GPT-4o	71.45	69.69	78.60	45.94	24.22
		GPT-3.5-turbo	68.87	67.80	80.42	46.05	25.73
		Mistral-7b-instruct	69.20	70.56	78.83	45.93	27.18
	text-embedding-3-large	-	72.52	72.86	83.81	54.14	26.25
		GPT-4o	75.57	72.24	83.40	54.33	25.49
		GPT-3.5-turbo	74.44	72.18	83.59	53.85	27.36
		Mistral-7b-instruct	73.97	72.44	83.30	54.51	26.99

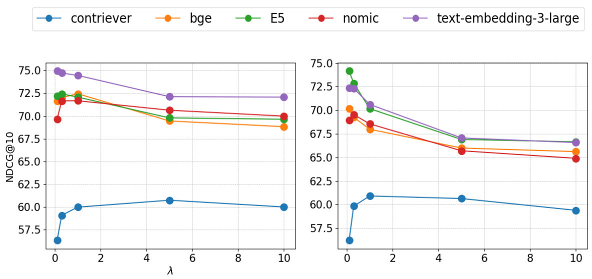
Table 1: NDCG@10 results produced by different retrievers, embedding models, and hypothetical query generators (LLMs) across various datasets. The '−' sign indicates that the results in the associated row are generated with the baseline embedding model. The light gray color indicates that using HyQE with all three LLMs outperforms the baseline embedding model for the associated dataset. The blue color indicates that the highest NDCG@10 value for a combination of retriever and embedding models under a dataset is achieved by HyQE. According to the MTEB leaderboard (Muennighoff et al., 2022), increasing NDCG@10 by 1 can improve the ranking by up to 10 positions.

which is a sparse retrieval model that does not generate text embeddings. We use the pre-built Lucene indexes in Pyserini (Lin et al., 2021) for retrieval. We use five embedding models as the baselines for ranking: contriever (Izacard et al., 2021), bge-base-en-v1.5 (Xiao et al., 2023), E5-large-v2 (Wang et al., 2022), text-embedding-3-large, and nomic-embed-text-v1.5 (Nussbaum et al., 2024). We also use those embedding models as the backbones of HyQE and compare the results produced by HyQE with those produced by the baseline embedding models. We use three different LLMs to generate the hypothetical queries: Mistral-7b-instruct-v0.2 (Jiang et al., 2023), GPT-3.5 turbo, and GPT-4o.

Implementation Details. We first retrieve 100 contexts with a retriever. Then, we use an embedding model to rank the contexts based on the cosine similarity between the context and the query and produce an ordered-set $C_{q,K}$ of candidate contexts where we set $K = 30$. Then, we use the proposed method to obtain r_q and re-rank these 30 contexts. Then, we compare these results with the ranking produced by the embedding model.

Main Results. Table 1 shows the NDCG@10 produced by our methods and baseline embedding models on the benchmarks. The Retrieval Model and Embedding Model columns indicate which models provide the initial list of 100 contexts and which model is used for providing the $C_{q,30}$ candidate contexts. The HyQE Model column indicates which LLMs are used to generate the hypothetical queries. The symbol '−' indicates that the results in the associated rows are produced by the baseline embedding models without hypothetical queries. The other rows are obtained by HyQE framework with different combinations of retrieval models, embedding models, and hypothetical query generators. Our methods outperform the associated baseline embedding models most of the time. These results answer our main question and Question A, showing that locally hosted small-sized models and closed-source proprietary large models can generate high-quality hypothetical queries that result in high-quality rankings in our framework.

Changing the hyperparameters. Next, we answer Question B by changing the hyperparameters λ in Eq.2 to examine how sensitive the HyQE framework is to the changes. We pick 2 datasets, 4 embedding models, i.e., contriever, bge-base-en-v1.5, E5-large-v2, and nomic-embed-text-v1.5, and use SPLADE++ ED as the retriever so that the candidate contexts are the same. Fig.4 shows that NDCG@10 decreases as λ increases for most embedding models, suggesting choosing small λ for these models. In Appendix C, we will present the results of modifying λ for other datasets, and we will also explore the impact of changing the number of candidate contexts, i.e., the K in $C_{q,K}$, from 30 to other values.



(a) DL19 (b) DL20

Figure 4: NDCG@10 changes with λ.

Compatibility with HyDE. To examine whether HyQE is compatible with other methods, we combine our method with HyDE (Gao et al., 2023a) by using HyDE for context retrieval and HyQE for context ranking. We use the identical embedding models for context retrieval and ranking, and use GPT-4o for the hypothetical context and query generation. Since HyDE generates hypothetical contexts and uses the average of the query embedding and hypothetical embeddings for context retrieval, we implement this combination in two ways. The first is to only use HyDE to collect 100 contexts and repeat the context ranking with HyQE as in Algorithm 1. The second is to use HyDE to not only collect the 100 contexts but also replace the query embedding with the mean of the query and hypothetical context embeddings during execution of Algorithm 1. In Table 2, we compare the results obtained in these two ways as well as those of using HyDE alone. The results answer Question C by showing that HyQE is not only compatible with

Embedding Model	HyDE	DL19	DL20	COVID	NEWS	Touche
contriever	-	62.60	57.69	53.86	38.76	17.92
	+HyQE	65.58	62.72	54.39	43.59	18.81
	×HyQE	67.38	63.35	57.52	45.49	20.41
bge-base-en-v1.5	-	75.37	70.55	75.49	43.55	17.92
	+HyQE	75.16	71.36	78.98	46.12	20.69
	×HyQE	75.96	72.07	78.81	46.85	20.39

Table 2: NDCG@10 results produced by combining HyDE with HyQE. In the 'HyDE' column, the '-' symbol indicates that the results in the associated rows are generated by HyDE; '+ HyQE' indicates that HyDE is used to retrieve contexts, but the query embedding is not changed when HyQE ranks the contexts; '×HyQE' indicates that the query embedding has been changed into the average embedding of the query and hypothetical contexts when HyQE ranks the contexts. The font color scheme is similar to that in Table 1.

Embedding Model	DL19	DL20	COVID	NEWS	Touche
contriever	51.33	46.76	33.10	38.87	15.33
bge-base-en-v1.5	71.04	66.48	79.52	43.57	18.40

Table 3: NDCG@10 results produced by using Eq.5 for HyQE.

HyDE but also can further improve the ranking quality beyond that in Table 1.

Using the Alternative Scoring Function. We next answer Question D by evaluating the alternative scoring function in Eq.5. We use two embedding models, i.e., contriever and bge-base-en-v1.5, for both context retrieval and ranking. The hyperparameter λ for each embedding model stays the same as that produces the main results. We still use GPT-4o to generate hypothetical queries. The results are included in Table 3. By comparing with the results in Table 1, it is obvious that using Eq.5 outperforms the baseline embedding models and cannot outperform using Eq.2.

7 Conclusion

In this paper, we introduce a novel framework for context ranking using hypothetical queries generated by LLMs. Our method is grounded in variational inference, aiming to preserve the causal relationship between queries and the contexts. The experimental results demonstrate that our approach not only outperforms baselines but also can be integrated seamlessly with existing techniques, allowing for iterative refinement and continuous improvement. Furthermore, our method can amortize the overhead in text generation with LLM as the input queries increase, offering a scalable and efficient solution for context retrieval and ranking.

8 limitation

While our proposed framework demonstrates significant improvements in context ranking and is scalable, there are several limitations to consider:

1. **Overhead of Query Generation and Storage.** The effectiveness of our method relies on using an LLM to generate the queries. The computational complexity for the query generation is amortized as the input queries grow. However, this amortization is built on the premise that the generated queries are stored for future retrieval. And such storage will raise the memory complexity of this framework. As a result, extremely large datasets could still pose challenges.

2. **Dependency on the Type of Query.** The input query can have different types, e.g., questions asking for specific information, a sequence of keywords, etc. However, in the prompt we only ask the LLM to generate the questions that can be addressed by the context, which may not have different structures than the input query.

3. **Adaptability to Context Chunk Sizes.** Our framework has been validated on well-known TREC and MS-MARCO datasets, where the contexts are provided. However, when dealing with document retrieval, the contexts are created by segmenting the documents into chunks. The documents may be segmented with different chunk sizes depending on the requirement. Each time the document is segmented, the hypothetical queries have to be regenerated from the contexts. This issue could potentially be mitigated by generating hypothetical queries from smaller, fixed-sized chunks of contexts and composing those queries for larger chunks of contexts. However, the specifics of this approach require further investigation to ensure its effectiveness and efficiency.

Addressing these limitations in future work will be essential for enhancing the robustness, efficiency, and applicability of our proposed context ranking framework across a broader range of scenarios.

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*.

Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. 2021. Invariance principle meets information bottleneck for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 34:3438–3450.

Tiago Almeida and Sérgio Matos. 2024. Exploring efficient zero-shot synthetic dataset generation for information retrieval. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1214–1231, St. Julian's, Malta. Association for Computational Linguistics.

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023a. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023b. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Akari Asai, Xinyan Yu, Jungo Kasai, and Hannaneh Hajishirzi. 2021. One question answering model for many languages with cross-lingual dense passage retrieval. In *Advances in Neural Information Processing Systems*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1).

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. 2019. Virel: A variational inference framework for reinforcement learning. *Advances in neural information processing systems*, 32.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From distillation to hard negative sampling: Making sparse neural ir models more effective. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 2353–2359.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023a. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. *CoRR*, abs/2112.09118.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

J. L. W. V. Jensen. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. 2023. Mistral 7b (2023). *arXiv preprint arXiv:2310.06825*.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

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.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 120–127, New York, NY, USA. Association for Computing Machinery.

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.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. 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)*, pages 2356–2362.

Linqing Liu, Minghan Li, Jimmy Lin, Sebastian Riedel, and Pontus Stenetorp. 2022. Query expansion using contextual clue sampling with language models. *arXiv preprint arXiv:2210.07093*.

Chaochao Lu, Yuhuai Wu, José Miguel Hernández-Lobato, and Bernhard Schölkopf. 2022. Invariant causal representation learning for out-of-distribution generalization. In *International Conference on Learning Representations*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane

Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Transactions on Machine Learning Research*. Survey Certification.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. *Preprint*, arXiv:2402.01613.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? In *Thirty-seventh Conference on Neural Information Processing Systems*.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. 2021. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*.

Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. 2024. Beyond the limits: A survey of techniques to extend the context length in large language models. *arXiv preprint arXiv:2402.02244*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Weichao Zhou and Wenchao Li. 2022. A hierarchical bayesian approach to inverse reinforcement learning with symbolic reward machines. In *International Conference on Machine Learning*, pages 27159–27178. PMLR.

A Visualizing the Hypothetical Query Emebeddings

We demonstrate the difference between the contexts ranked by cosine similarity and those by HyQE. We conduct an independent component analysis (ICA) on each high-dimensional text embedding and project the embeddings onto a 2-D plane, using the two principal dimensions as the axes.

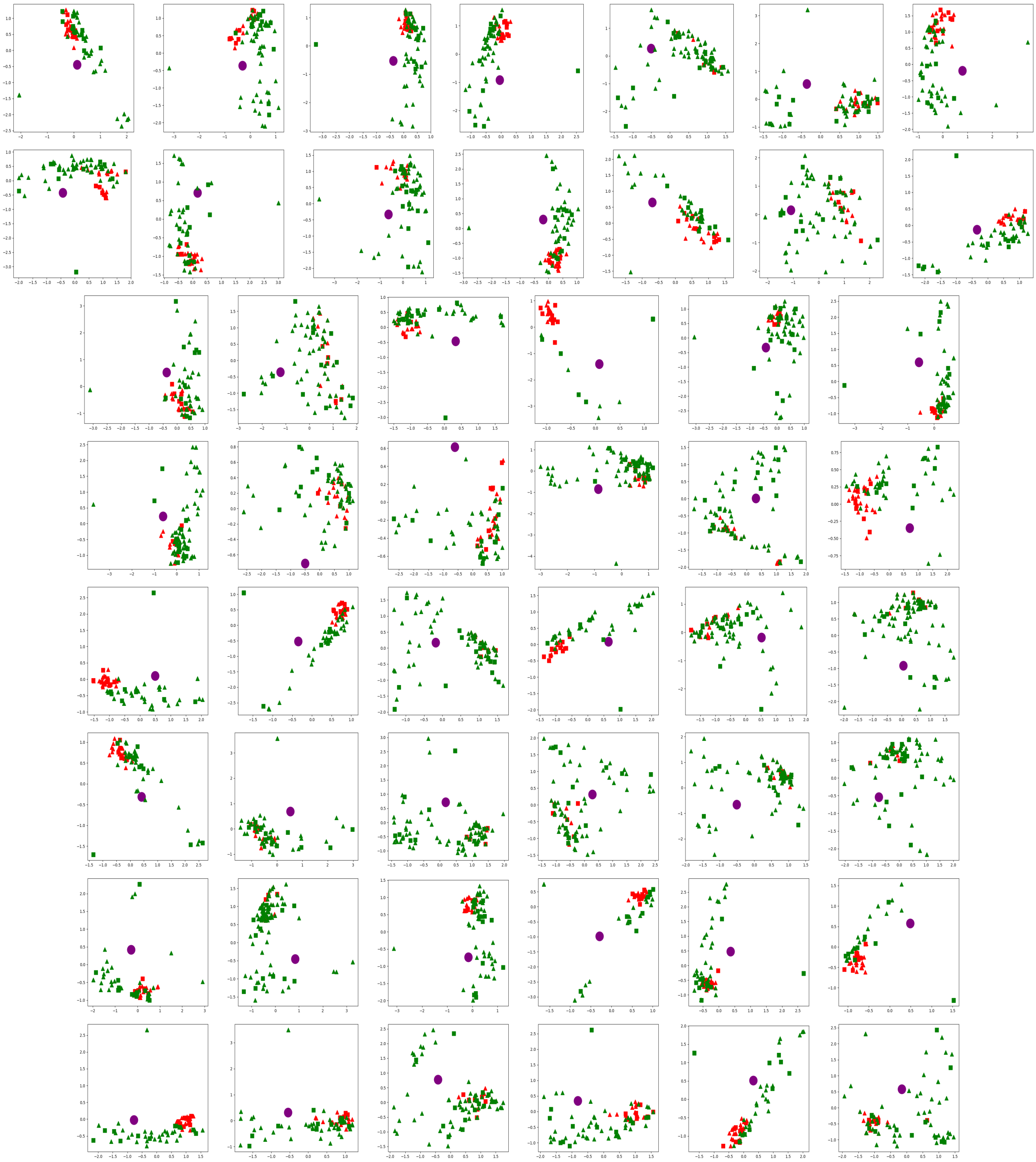


Figure 5: ICA on the bge-base-env-v1.5 embeddings for the COVID dataset, which contains 50 input queries. Each figure corresponds to one of the input queries. The purple circles represent the queries. The red squares represent the top 5 contexts ranked using cosine similarity, and the red triangles represent the corresponding hypothetical queries. The green squares represent the top 5 contexts ranked using our method, and the green triangles represent the corresponding hypothetical queries.

It can be observed from Fig.5 that the contexts ranked by cosine similarity tend to cluster near the input query in the embedding space. In contrast, the contexts ranked by HyQE and their corresponding hypothetical queries are more scattered. This suggests that, in the embedding space, the queries are not necessarily adjacent to the contexts that provide answers to them. Our experimental results in Table.1 show that the ranking produced by our HyQE has a higher NDCG@10 value than that of cosine similarity. Therefore, both the ICA visualization and the evaluation results support our proposition that cosine similarity should be applied only when comparing queries with queries to ensure better preservation of the causal structure and to avoid spurious correlations.

B Additional Implementation Details

In our implementation, we have used Mistral-7b-instruct-v0.2, GPT-3.5-turbo, and GPT-4o to generate hypothetical queries.

For Mistral-7b-instruct-v0.2, we use the pre-trained model. We set the context window size as 3900, and the maximum number of outputs as 1024. We also use an instruction prompt as shown in Fig.6 to wrap the prompt in Fig.2.

```
<s>[INST]\nYou are an AI assistant. Here are some rules you always follow:
- Generate human readable output, avoid creating output with gibberish text.
- Don't plainly replicate the given instruction.
- Generate only the requested output, don't include any other language before or
    after the requested output.
- Never say thank you, that you are happy to help, that you are an AI agent, etc.
    Just answer directly.
- Generate professional language typically used in business documents in North
    America.
- Never generate offensive or foul language.

The user prompt is as follows:\n\n\{prompt}[/INST]</s>
```

Figure 6: Instruction Prompt for Mistral-7b-instruct-v0.2. '{prompt}' is the placeholder for the prompt shown in Fig.2.

We show examples of the hypothetical queries generated by Mistral-7b-instruct-v0.2 in Fig.7.

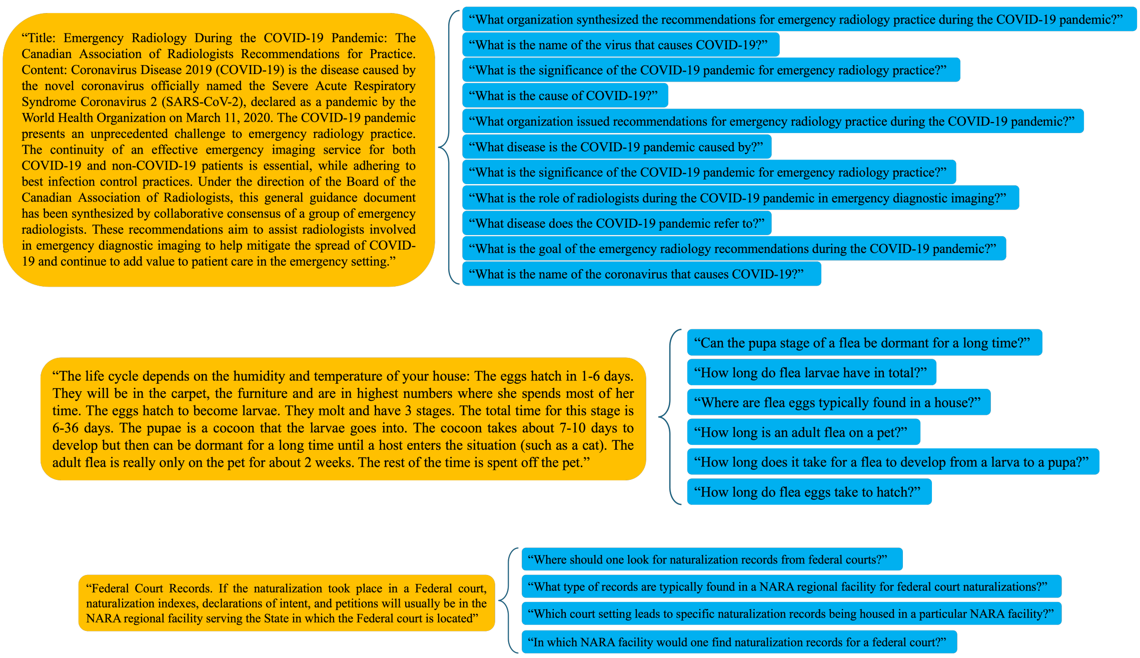


Figure 7: Contexts and the corresponding hypothetical queries generated by Mistral-7b-instruct-v0.2. The contexts are in the yellow bubble. The hypothetical queries are in the blue bubbles.

For GPT-3.5-turbo and GPT-4o, we send the following message to OpenAI API with the parameters $\texttt{temperature} = 0.1, \texttt{top\_k} = 1$ and $\texttt{n} = 1$ in the request. For the same contexts in Fig.7, GPT-4o generates the queries as shown in Fig.9.

```
{
    "role": "system",
    "content": "
        You are an AI assistant. Here are some rules you always follow:
        - Generate human readable output, avoid creating output with gibberish text.
        - Don't plainly replicate the given instruction.
        - Generate only the requested output, don't include any other language
          before or after the requested output.
        - Never say thank you, that you are happy to help, that you are an AI agent,
           etc. Just answer directly.
        - Generate professional language typically used in business documents in
          North America.
        - Never generate offensive or foul language,
        "
},
{
    "role": "user",
    "content": {prompt},
}
```

Figure 8: Messages sent to OpenAI API. '{prompt}' is the placeholder for the prompt shown in Fig.2.

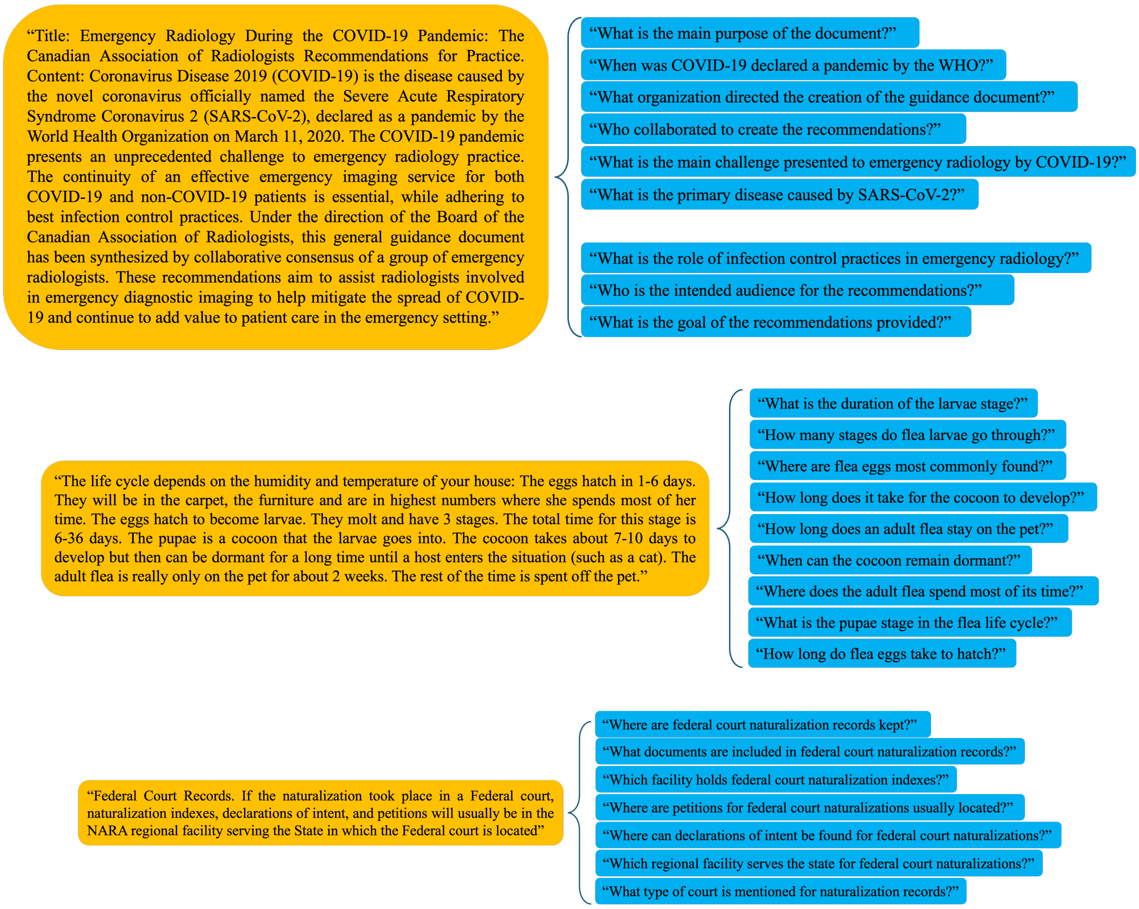


Figure 9: Contexts and the corresponding hypothetical queries generated by GPT-4o. The contexts are in the yellow bubble. The hypothetical queries are in the blue bubbles.

We mentioned in Section 6 that we use a different prompt from that in Fig.2 for the Touche dataset. The prompt is shown in Fig.10. We designed this prompt because each query in this dataset is about the topic of an argument, and the contexts record the dialogues in the argument, which may deviate from the topic. An example is provided in Fig.1.

In Table 4 we show the hyperparameter λ we set for each embedding model to obtain the results in

```
Which topics could the 'Content' section of the following passage be arguing about.
If the 'Content' section provides no meaningful argument, respond with a single 'No
    content'.

```<passage>
{context}
</passage>```

Topics are questions.
Each question must be very short, different, and be written on separate lines.
Do not mention the passage itself or the author of the passage...
```

Figure 10: Prompt designed for the Touche2020 dataset. '{context}' is the placeholder for the context.

Table 1. Note that for bge-base-env-v1.5, we use a much smaller $\lambda$ than other models because we do not normalize the product between the embeddings of the input queries and hypothetical queries but normalize the product between the embeddings of the queries and contexts. In this way, we obtain better and more stable results than those when we normalize all the products.

| | contriever | bge-base-en-v1.5 | E5-large-v2 | nomic-embed-text-v1.5 | text-embedding-3-large |
|---|---|---|---|---|---|
| $\lambda$ | 2.0 | 0.03 | 0.5 | 0.5 | 0.3 |

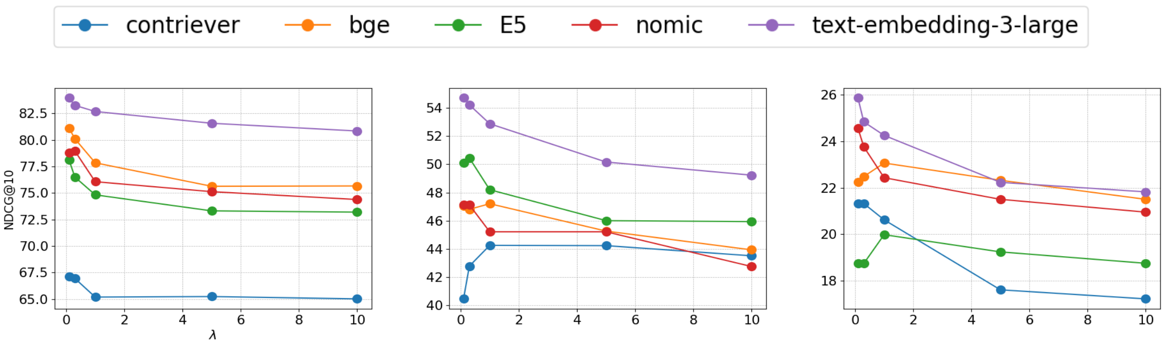
Table 4: Hyperparameter $\lambda$ used for each embedding model to produce results in Table 1.

Next, we show the derivation of ELBO in Eq.3.

$$
\begin{aligned}
& D_{KL}(p_q(c)||p(c|q)) \\
= \; & \mathbb{E}_{c\sim p_q(c)}[\log p_q(c) - \log p(c|q)] \\
= \; & \mathbb{E}_{c\sim p_q(c)}[\log p_q(c) - \log \frac{p(q|c)p(c)}{p(q)}] \\
= \; & \mathbb{E}_{c\sim p_q(c)}[\log p_q(c) - \log p(c)] - \mathbb{E}_{c\sim p_q(c)}[\log p(q|c)] + \mathbb{E}_{c\sim p_q(c)}[\log p(q)] \\
= \; & D_{KL}(p_q(c)||p(c)) - \mathbb{E}_{c\sim p_q(c)}[\log p(q|c)] + \log p(q) \\
\leq \; & ELBO
\end{aligned}
$$

# C Additional Experimental Results

In Section 6, we have shown how changing the hyperparameter $\lambda$ affects HyQE on the DL19 and DL20 datasets. We now show the results on 3 other datasets. Most of the results align with those in the main text, suggesting choosing a small $\lambda$ for all models except for contriever.




(a) COVID

(b) NEWS

(c) Touche

Figure 11: NDCG@10 changes with $\lambda$.

We have also tried to use different embedding models for retrieval and ranking. As shown in Table
```

5, the results align with those reported in the main text, indicating that HyQE can enhance the ranking quality.

Retrieval Model	Embedding Model	HyQE Model	DL19	DL20	COVID	NEWS
contriever	bge-base-en-v1.5	-	65.52	62.29	51.60	42.59
		GPT-3.5-turbo	66.16	62.15	53.80	42.69
	E5-large-v2	-	66.24	65.20	47.08	46.72
		GPT-3.5-turbo	66.44	64.94	51.51	47.17
	nomic-embed-text-v1.5	-	63.27	60.07	54.07	43.34
		GPT-3.5-turbo	64.52	62.09	53.39	44.20
bge-base-en-v1.5	contriever	-	52.78	51.10	63.57	40.17
		GPT-3.5-turbo	59.56	56.73	73.62	45.47
	E5-large-v2	-	69.48	71.01	66.19	48.10
		GPT-3.5-turbo	71.92	71.36	77.62	48.41
	nomic-embed-text-v1.5	-	68.20	65.61	77.25	43.60
		GPT-3.5-turbo	71.28	67.20	77.69	44.30

Table 5: NDCG@10 results produced by different combinations of embedding models across various datasets. The '−' sign indicates that the results in the associated row are generated without HyQE. The blue color highlights that using HyQE for ranking results in a higher NDCG@10 value compared to not using HyQE for the combination of embedding models and dataset.

In Algorithm 1, the parameter K in the candidate context set $C_{q,K}$ functions can also be considered as a hyperparameter. Setting a small value for K limits the range of contexts to be ranked, resulting in fewer calls to the LLM. Conversely, a large value of K allows for low-rank but potentially highly relevant contexts to be re-ranked. However, this increases the number of calls to the LLM and the risk of erroneously assigning a high rank to a low-relevant context. In Section 6, the results are obtained with K set to 30. In Table 6, we show how the performance of HyQE changes with the value of K. Compared with Table 1, the results for $K = 20$ and $K = 30$ are close to each other.

Retrieval Model	Embedding Model	HyQE Model	K	DL19	DL20	COVID	NEWS
contriever	contriever	GPT-3.5-turbo	10	46.35	43.56	28.84	36.74
			20	51.38	48.59	32.82	41.33
		Mistral-7b-instruct	10	46.14	42.94	28.63	36.24
			20	50.85	48.16	32.90	40.18
	bge-base-en-v1.5	GPT-3.5-turbo	10	66.55	61.84	52.66	42.09
			20	66.16	62.15	53.80	42.69
		Mistral-7b-instruct	10	66.58	61.94	52.62	42.31
			20	65.89	62.40	53.93	42.99
	E5-large-v2	GPT-3.5-turbo	10	66.55	64.85	27.32	34.84
			20	66.48	64.98	27.32	34.84
		Mistral-7b-instruct	10	66.41	64.84	48.38	46.08
			20	65.67	65.09	52.78	46.28
	nomic-embed-text-v1.5	GPT-3.5-turbo	10	62.12	61.77	53.92	44.83
			20	64.11	62.08	53.26	43.86
		Mistral-7b-instruct	10	64.48	61.54	55.46	43.09
			20	63.47	63.69	54.71	44.22

Table 6: NDCG@10 results produced by embedding models and hypothetical query generators (LLMs) across various datasets. The values in the K column indicates HyQE is used to re-rank the top-K contexts ordered by the embedding model.