Database-Augmented Query Representation for Information Retrieval

Anonymous ACL submission

Abstract

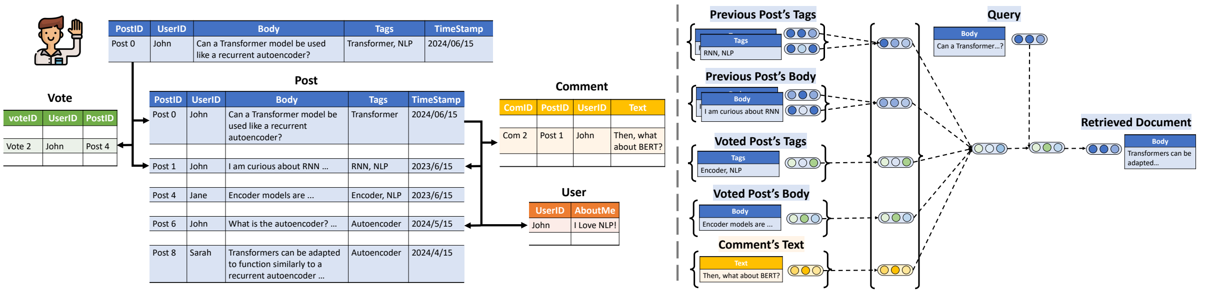
Information retrieval models that aim to search for the documents relevant to the given query have shown many successes, which have been applied to diverse tasks. However, the query provided by the user is oftentimes very short, which challenges the retrievers to correctly fetch relevant documents. To tackle this, existing studies have proposed expanding the query with a couple of additional (user-related) features related to the query. Yet, they may be suboptimal to effectively augment the query, though there is plenty of information available to augment it in a relational database. Motivated by this, we present a novel retrieval framework called Database-Augmented Query representation (DAQu), which augments the original query with various (query-related) metadata across multiple tables. In addition, as the number of features in the metadata can be very large and there is no order among them, we encode them with our graph-based set encoding strategy, which considers hierarchies of features in the database without order. We validate DAQu in diverse retrieval scenarios that can incorporate metadata from the relational database, demonstrating that ours significantly enhances overall retrieval performance, compared to existing query augmentation methods.

1 Introduction

Information Retrieval (IR) is the task of fetching query-relevant documents from a large corpus. Traditional approaches have focused on sparse retrieval, which searches for documents that yield the highest lexical match with the given query (Robertson et al., 1994). Recently, advancements in neural language models have led to the introduction of dense retrieval models, which represent both the query and the document in a learnable latent space and then calculate their similarity on it (Karpukhin et al., 2020; Izacard et al., 2022). Notably, these IR systems have gained much attention in the era of Large Language Models (LLMs), due to their ability to assist LLMs help generating accurate answers with evolving knowledge from an external source, which is particularly valuable as LLMs are intrinsically vulnerable to problems of hallucination and maintaining up-to-date knowledge (Cho et al., 2023; Ding et al., 2024; Jeong et al., 2024).

Despite such a huge advantage of IR in NLP, it faces a critical challenge that information captured in a query itself is oftentimes not sufficient to retrieve its relevant documents from the external corpus, due to the scarcity of information within its (shorter) text. To overcome this challenge, previous work has focused on enriching representations of queries or documents by expanding them with additional texts or augmenting their representation spaces (Jeong et al., 2022; Jagerman et al., 2023; Lin et al., 2023a). However, despite their improvement, those previous approaches are still limited in that they rely on the capability of models (e.g., LLMs) used during augmentation, though there can be external knowledge sources (for augmentation) that are associated with the user query (such as the user's purchase history for shopping-related queries). While some other work has considered these additional sources, enhancing the representation of queries with them, they leverage only a single source of information stores, especially the one specific to the user (who issues the query) (Gupta et al., 2019; Zhang et al., 2020; Deng et al., 2021; Buss et al., 2023). However, in the real world, data (including queries) is usually mapped into the database and linked to other data within it, which means that plenty of information that can be potentially used for query enrichment is available on the relational database (Fey et al., 2023).

Therefore, in this work, we introduce a novel IR paradigm, Data-Augmented Query representation (DAQu), which augments representations of queries by searching for and connecting their associated information across multiple relational tables



PostID	UserID	Body	Tags	TimeStamp
Post 0	John	Can a Transformer model be used like a recurrent autoencoder?	Transformer, NLP	2024/06/15

Post

PostID	UserID	Body	Tags	TimeStamp
Post 0	John	Can a Transformer model be used like a recurrent autoencoder?	Transformer	2024/06/15
Post 1	John	I am curious about RNN …	RNN, NLP	2023/6/15
Post 4	Jane	Encoder models are …	Encoder, NLP	2023/6/15
Post 6	John	What is the autoencoder? …	Autoencoder	2024/5/15
Post 8	Sarah	Transformers can be adapted to function similarly to a recurrent autoencoder …	Autoencoder	2024/4/15

Vote

voteID	UserID	PostID
Vote 2	John	Post 4

Comment

ComID	PostID	UserID	Text
Com 2	Post 1	John	Then, what about BERT?

User

UserID	AboutMe
John	I Love NLP!

Figure 1: A conceptual illustration of our proposed DAQu, which particularly denotes a link among multiple tables for the given query (Left) and a graph-based set-encoding strategy that encodes metadata hierarchically (Right).

within the database. Specifically, as shown in Figure 1, for the task of retrieving relevant answers to the given question for the Stack Exchange dataset, we represent the query with its own information but also its relevant information within and across the multiple tables, such as its title, body, and tags in the same table but also its poster's previous posts, answers (that they like), bios, and badges (earned) spread over other tables. However, the volume of these metadata can be extremely large, and simply expanding the query with additional terms in the metadata (as done in existing query expansion work (Gupta et al., 2019; Deng et al., 2021)) is not feasible due to the limited context length of LMs. Moreover, since there is no inherent order for the elements in the metadata, the query augmentation approach should ensure order invariance when incorporating these diverse pieces of information.

To this end, we further propose to encode various query-related metadata within and across multiple tables over the relational database, based on a graph set encoding scheme. Specifically, there are multiple columns associated with the given query (within and across different tables), and each of these columns further has multiple query-related elements (such as previous posts made by the user who issues the query). Thus, to effectively represent these relational metadata, we first aggregate query-related cells for each column into one representation, and then aggregate representations of all columns (aggregated from their corresponding cells) into another representation. Then, this final representation can be viewed as the representation for query-related metadata, which can then be used for augmenting the representation of the query. It is worth noting that those two layer structures (aggregation on column- and query-level) can be viewed as a two-layer graph neural network (Kipf and Welling, 2017; Gilmer et al., 2017) since each layer captures the interactions between the nodes (in this case, cells and then columns) hierarchically.

We validate our DAQu on retrieval tasks designed with the Stack Exchange and the Amazon Product Catalog databases from Fey et al. (2023). The experimental results show significant improvements of our DAQu in retrieval performance compared to other query augmentation baselines across diverse scenarios. Moreover, we demonstrate that the graph set encoding technique operationalized in our DAQu effectively represents metadata, enhancing the representations of queries for retrieval.

Our contributions and findings are threefold:

- We present a new query augmentation paradigm for retrieval, which augments the query representation based on its relevant information linked to multiple tables over the relational database.
- To represent a large number of elements in the database with order invariance for query augmentation, we propose a graph set encoding approach that hierarchically represents them without order.
- We demonstrate the efficacy of DAQu on multiple retrieval scenarios designed with real-world databases against query augmentation baselines.

2 Related Work

Retrieval In response to a query from a user, the retrieval task is to search for the most relevant documents from a large corpus (such as Wikipedia) (Zhu et al., 2021). Typically, it can be performed with two types of models: sparse and dense retrievers. Specifically, sparse retrievers such as TF-IDF or BM25 (Robertson et al., 1994) represent the query and document based on their terms and frequencies in a sparse vector space, whereas dense retrievers use a trainable dense vector space to embed the query and document usually with language models (Karpukhin et al., 2020; Izacard et al., 2022). Recently, due to the limitation of sparse retrievers that are vulnerable to the vocabulary mismatch problem (where the retrieval fails when the lexical terms within the query and document are different), dense retrieval is widely selected as a default choice and many advancements have been made on it. For example, DPR (Karpukhin et al., 2020) is a supervised dense retriever with a dual-encoder architecture that is trained discriminatively on the labeled pair of a query and its relevant documents to achieve higher similarity scores than the pair of the query-irrelevant documents. Also, Contriever (Izacard et al., 2022) utilizes a self-supervised learning strategy, which generates its training samples by creating positive pairs from query-related contexts within and across documents, rather than relying on explicitly annotated data. Yet, using only the information within a query for retrieval can be sub-optimal, due to the scarcity of information on it.

Query Augmentation for Retrieval Some studies have proposed augmenting (or expanding) the original query with additional information to enhance the retrieval performance (Carpineto and Romano, 2012; Azad and Deepak, 2019). To be specific, traditional query augmentation methods have focused on utilizing a lexical knowledge base such as the WordNet (Miller, 1992) to expand the original queries (Bhogal et al., 2007; Zhang et al., 2009). In addition, some other work has implemented statistical models such as RM3 (Jaleel et al., 2004a), which add new terms to the query extracted from the top documents in the initial search results and then adjust their weights based on their importance (Lavrenko and Croft, 2001; Jaleel et al., 2004b; Lv and Zhai, 2009). However, these methods have been shown to be not very effective and, in some cases, even degraded the retrieval performance (Nogueira et al., 2019; Jeong et al., 2021). Therefore, recent work has turned to leveraging neural models to extract or generate query-relevant terms and then append such terms to the original query (Esposito et al., 2020; Zheng et al., 2020; Mao et al., 2021). Moreover, further advances have been made by incorporating recent LLMs to utilize their remarkable capabilities in generating such terms (Wang et al., 2023; Shao et al., 2023; Buss et al., 2023; Jagerman et al., 2023; Feng et al., 2024; Dhole and Agichtein, 2024). However, despite the fact that the query is represented and leveraged on the latent space with the recent dense retrievers, existing work focuses on explicitly expanding its text (instead of manipulating this query representation for augmentation). This approach may be problematic if there is a significant amount of data available to augment the query representation across multiple relational tables over the database.

Retrieval with Database A natural way to store a collection of data is to use a relational database, that is designed to effectively manage, retrieve, and manipulate (up-to-date) data for various applications (Johnson et al., 2016; Fey et al., 2023). Recently, to utilize the data in the database for question answering, the task of retrieving the tabular structures and the information in them has increasingly gained much attention. To be specific, some studies have developed the approach to retrieve the tables themselves (relevant to the given query) from a large table corpus (Herzig et al., 2021; Wang et al., 2022). In addition, some other work extends this approach, extracting or generating the answer for the query from the retrieved tables (Pan et al., 2021, 2022; Lin et al., 2023b). However, since some real-world questions require multiple tables, more recent studies have made further progress, thus proposing to incorporate multiple tables during retrieval (Kweon et al., 2023; Chen et al., 2024) or reading the tables (Pal et al., 2023). However, unlike all the aforementioned work that has focused on retrieving the tables themselves and finding relevant cells within them, our work is completely different, which aims to effectively handle the query for document retrieval by using the query-related information spread across multiple tables, to augment the representation of the query.

3 Method

In this section, we describe our method of augmenting the representation of the query for IR with the information stored within the relational database.

3.1 Preliminaries

We begin with preliminaries, providing formal descriptions of the retrieval and query reformulation based on representation-level augmentation.

Dense Retrieval Let us define the given query from a user as q and its relevant document as $d \in \mathcal{D}$, where $\mathcal{D}$ is an external document corpus. Then, to operationalize retrieval, we should be able to calculate the similarity between the query q and the document d, as $f(q, d)$, where f is a scoring function. Following the bi-encoder architecture for dense retrieval, in this work, we obtain the similarity by representing the query and document with encoders Enc_q and Enc_d parameterized by θ_q and θ_d, respectively, formalized as follows:

$$\begin{aligned} & f(q, d) = \text{sim}(\boldsymbol{q}, \boldsymbol{d}), \\ & \boldsymbol{q} = \mathsf{Enc}_q(q; \theta_q) \quad \text{and} \quad \boldsymbol{d} = \mathsf{Enc}_d(d; \theta_d), \end{aligned} \tag{1}$$

where $\boldsymbol{q}$ is the query representation and $\boldsymbol{d}$ is the document representation on the latent space. In addition, sim is a similarity metric, which is typically either cosine similarity or dot product.

It is worth noting that the objective of the dense retrieval function f is to rank the pair of the query q and its relevant document d^+ highest among all the other pairs with irrelevant documents $\{d_i^-\}_{i=1}^N$. To reflect this and following recent dense retrievers (Karpukhin et al., 2020; Izacard et al., 2022), we formalize the training objective, as follows:

$$l = -\log \frac{e^{f(q,d^+)}}{e^{f(q,d^+)} + \sum_{i=1}^{N} e^{f(q,d_i{}^-)}}. \tag{2}$$

Query Augmentation for Retrieval To improve the effectiveness of the dense retrieval (while tackling the limited contextual information within the query q), the textual query itself or its representation $\boldsymbol{q}$ can be enriched by augmenting it with the information that is not present in the original q but is crucial for minimizing the retrieval loss l. In this work, to effectively incorporate diverse pieces of information into the query without their order variance, we turn to augmenting the query representation $\boldsymbol{q}$ over the latent space, which is represented as follows:

$$\tilde{\boldsymbol{q}} = \lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{q}', \tag{3}$$

where $\tilde{\boldsymbol{q}}$ is the reformulated query representation, $\boldsymbol{q}'$ is the representation of the additional information helpful to enrich the query representation, and $\lambda \in [0, 1]$ is for giving weight to it.

3.2 Database-Augmented Query Representation

We now introduce our database-augmented query representation framework for information retrieval.

Relational Database It is noted that a vast amount of valuable information (in the real world) is typically stored in a relational database, and, inspired by this, we aim to augment the representations of queries with the relevant information within this relational database. In this paragraph, we first provide its general description. Formally, the relational database is defined as a set of tables: $\mathcal{T} = \{T_i\}_{i=1}^N$, and each table is comprised of a collection of rows $T = \{r_j\}_{j=1}^K$, where N is the number of tables and K is the number of rows.

We note that one of the valuable characteristics of the relational database is that some rows in tables are connected with others in other tables, which facilitates relational linkages and ease of data retrieval. Formally, each row r_i in the table consists of a primary key column that uniquely identifies each row within the table, (potentially) some foreign key columns that link to primary keys in other tables, and other non-key attribute columns providing additional information about the row. In other words, the relationships between primary and foreign keys connect rows across different tables, and other attribute columns store descriptive information about the rows. Formally, if a foreign key column f in table T_i references a primary key column p in T_j, we can represent their relationship as (f_i, p_j). In addition, all such relationships between all different tables can be denoted as $\mathcal{L} = \{(f_i, p_j)\}_{(i,j)}$ where $\mathcal{L} \subseteq T \times T$.

For example, analogous to the Amazon database, let's assume that the table T_{review} includes the primary key column REVIEWID, the foreign key column PRODUCTID, and the attribute column TEXT. Also, the table $T_{product}$ has the primary key column PRODUCTID and the attribute column DESCRIPTION. Lastly, the foreign key column PRODUCTID in T_{review} points to the primary key column in $T_{product}$. Then, the relationships between those two tables can be represented with a pair of primary and foreign keys: $(\text{PRODUCTID}_{review}, \text{PRODUCTID}_{product})$.

Query Augmentation with Relational Database Recall that the equation to augment the representation of the given query is formalized as $\tilde{\boldsymbol{q}} = \lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{q}'$. We note that, in this work, $\boldsymbol{q}'$ is the representation that we obtain from the query-related information within the relational database, and we now turn to explain how to get $\boldsymbol{q}'$.

Formally, each query that the user requests can be considered as one row r_j in a certain table T_i. For example, in the Stack Exchange dataset, the query that the user posts is stored in the table as one row: $r \in T_{post}$, where this row (query) r consists of the primary key (POSTID), the foreign key (USERID), and the multiple attributes (such as BODY, TAGS, and TIMESTAMP). Then, based on the following relational structure of this database:

$$\begin{aligned}\mathcal{L} = \{&(\text{USERID}_{user}, \text{USERID}_{post}),\\ &(\text{USERID}_{vote}, \text{USERID}_{post}),\\ &(\text{POSTID}_{post}, \text{POSTID}_{comment}), \ldots\},\end{aligned} \tag{4}$$

the row for the query in the post table can be linked to other rows in different tables, for example, the user table, vote table, and comment table connected with USERID and POSTID columns (Figure 1).

Note that this relational structure of the database allows us to utilize diverse pieces of information (within the same and across different tables) when enriching the query representation $\boldsymbol{q}$. Specifically, to represent the embedding for query metadata $\boldsymbol{q}'$ (used for augmenting the original query representation $\boldsymbol{q}$), we can not only use the attributes within the columns of the row for the query (such as BODY and TAGS of the post table T_{post}) but also the attributes of associated rows (to the query) from different tables (such as ABOUTME of the user table T_{user} associated with the column USERID).

Formally, we represent all the attributes of the rows associated with the given query (q) as follows:

$$\begin{aligned}\mathcal{A} = &\{r_{i,j} \mid r_i = q\} \cup \\ &\{r_{i,j} \mid q \in T \text{ and } r_i \in T' \text{ and } (T, T') \in \mathcal{L}\} \cup \\ &\{r_{i,j} \mid r_i \in T \text{ and } q \in T' \text{ and } (T, T') \in \mathcal{L}\},\end{aligned} \quad (5)$$

where $r_{i,j}$ is the value of the jth attribute column of the ith row. Then, based on these attributes (the metadata), we derive their representation $\boldsymbol{q}'$ with the encoder: $\boldsymbol{q}' = \texttt{Enc}_a(\mathcal{A}; \theta_a)$, described below.

Graph-Structured Set Encoding We now turn to explain how to operationalize the encoding function $\texttt{Enc}_a(\cdot)$, which should effectively represent the diverse attributes $\mathcal{A}$ (over the relational database) into $\boldsymbol{q}'$, to enrich the original query representation $\boldsymbol{q}$ (as in Equation 3). To accomplish this objective, one possible strategy is to concatenate all the attribute values, and then encode the concatenated value with the encoder or append it to the original query (before encoding), following the existing query expansion work (Zheng et al., 2020; Deng et al., 2021; Dhole and Agichtein, 2024). However, despite their simplicity, these naïve expansion approaches have a couple of critical limitations. First, due to the large volume of data in the database, the number of attributes related to the query could be quite large, and it might be infeasible to encode their concatenated text with the encoder (due to its limited context length). In addition, the attributes do not have an inherent order (i.e., permutation invariant), making it arbitrary to determine the sequence in which they should be concatenated for encoding.

To tackle these challenges, in this work, we propose to consider attributes as the graph-structured set and subsequently encode them with the graph-structured set encoding strategy, which differs from and indeed extends the previous set encoding approach (Zaheer et al., 2017). Specifically, we first encode every attribute value $r_{i,j}$ in $\mathcal{A}$ into $\boldsymbol{r}_{i,j}$ with an attribute encoder: $\boldsymbol{r}_{i,j} = \texttt{Enc}_r(r_{i,j}; \theta_r)$, and then aggregate a group of encoded attributes according to each column into the single representation with mean pooling as $\boldsymbol{R}_j = \texttt{MEAN}(\{\boldsymbol{r}_{i,j}\}_{i=1})$, which then captures the representation of each category (or column) of the metadata. After that, we aggregate all these categorical (column-wise) representations into another representation, which represents the overall metadata for the given query as $\boldsymbol{q}' = \texttt{MEAN}(\{\boldsymbol{R}_j\}_{j=1})$. Note that this dual-layer structure — aggregating at both the column and query levels — resembles a two-layer graph neural network (Kipf and Welling, 2017; Gilmer et al., 2017), where each layer functionally captures the interactions between the attributes in the same column first and the columns over different tables next in a hierarchical manner.

For example, consider the scenario illustrated in Figure 1, where we aim to retrieve the answer post that the user selected as the best from the user query. Recall that, based on our formulation in Equation 3, its description is used for obtaining the query representation $\boldsymbol{q}$ and we enrich its representation with the representation from its metadata $\boldsymbol{q}'$, which we obtain from the proposed graph-structured set encoding. Specifically, the attributes $\mathcal{A}$ (metadata) include the comments (COMMENT) that the user previously wrote, and we encode them with the set encoding, formalized as $\boldsymbol{R}_{\text{COMMENT}} = \texttt{MEAN}(\{\texttt{Enc}_r(r_{i,\text{COMMENT}})\}_{i=1})$. Similarly, by extending this approach to other metadata categories, such as the previous tags from the posts the user wrote (TAGS) and the user profile (ABOUTME), we obtain their category-level representations as $\boldsymbol{R}_{\text{TAGS}}$ and $\boldsymbol{R}_{\text{ABOUTME}}$. After that, as a last step, we aggregate all the category-level representations into one single (comprehensive) query-level representation, formalized as follows: $\boldsymbol{q}' = \texttt{MEAN}(\{\boldsymbol{R}_{\text{COMMENT}}, \boldsymbol{R}_{\text{TAGS}}, \boldsymbol{R}_{\text{ABOUTME}}\})$, which is then used to augment the original query representation according to Equation 3.

Efficient Training Strategy with Metadata It should be noted that the number of attributes collected from the relational database is sometimes very large for certain queries, and it may be largely inefficient to consider all of them during training. To address this, we introduce a two-stage sample selection strategy to efficiently train a metadata encoder Enc_r and to efficiently obtain a metadata representation $\boldsymbol{q}'$. Specifically, due to the constraint on the GPU memory, it may not be possible to use all the attributes in $\mathcal{A}$ for parameter updates; therefore, during training, we randomly sample three attributes for each column and use only them to train the metadata encoder. In addition, while we can use all the remaining attributes (without gradients) to obtain the metadata representation along with the representations of three specific attributes for each column (with gradients), using all the remaining attributes may still be time-consuming and may yield the over-fitting issue; therefore, we randomly sample some of them and use only them to obtain the representation $\boldsymbol{q}'$. Meanwhile, in the inference step, we can utilize all the metadata attributes.

4 Experimental Setups

In this section, we describe the experimental setup, leaving further details in Appendix A.

4.1 Datasets

Since this is the first work on retrieval that utilizes the relational database for augmenting query representations, we design three novel retrieval tasks. Specifically, we construct two tasks with the Stack Exchange database and one task with the Amazon Product Catalog database from Fey et al. (2023).

Stack Exchange This dataset is collected from discussions in Stack Exchange[1], an online website for question-and-answering. All the information in this dataset is organized into the relational database, which consists of seven different tables (such as posts, users, and votes). In this work, based on this dataset, we design two retrieval tasks, as follows: **1) Answer Retrieval (Any Answer)** involves retrieving any answer posts made by other users in response to a specific question post. **2) Best Answer Retrieval (Best Answer)** is a more challenging task that aims to retrieve a single answer post that has been selected by the owner of the question post. In addition to those two retrieval tasks, we further consider two different scenarios by dividing the entire dataset by users (**SplitByUser**) or timestamps (**SplitByTime**). Specifically, for the first setting, the training, validation, and test sets are divided by users; therefore, there are no overlaps about users across these three subsets. Similarly, the later setting splits the dataset according to the timestamp that the post was made. Note that, for each retrieval instance, the information before the post timestamp is used to augment the query representation.

Amazon Product Catalog This dataset is collected from book reviews on the Amazon Product Catalog, which consists of three tables (such as users, products, and reviews) over the relational database. For this dataset, we introduce **3) Future Purchase Retrieval (Future Purchase)** as the retrieval task, which aims to predict any future book purchases of customers based on their current reviews as well as their previous purchases and reviews. Also, we construct two different settings for it, namely **ReviewToProduct** and **ProductToProduct**, where the first one uses the review text as a query while the latter one uses the product description as a query for retrieving future products.

4.2 Models

We explain the backbone retrieval models and the query augmentation baselines that we compare.

Retrieval Models We operationalize query augmentation approaches with two widely used dense retrieval models, namely DPR and Contriever, as follows: **DPR** is a supervised dense retrieval model that requires a pair of a query and its relevant document for training (Karpukhin et al., 2020); **Contriever** is another widely used dense retriever, but is trained in an unsupervised fashion (Izacard et al., 2022). In addition, as an indicator, we report the performance of the sparse retriever (**BM25**).

Augmentation Models We compare our DAQu against relevant query augmentation models as follows: **1) No Expansion (No Expan.):** This model directly uses the given query for retrieval without expanding it. **2) Naïve Query Expansion (Naïve Expan.):** This baseline concatenates a given query with all the textual terms of the associated metadata from the database. **3) Query Expansion w/ BM25 (Expan. w/ BM25):** Similar to Deng et al. (2021), this model also appends the metadata terms to the given query. However, before expanding the query, it employs a BM25 model to select metadata terms that are most relevant to the query, and only these selected terms are appended. **4) DAQu (Ours):** This is our model that augments the query representation by incorporating the metadata representation on a latent space, which is generated with the graph-structured set encoding strategy.

---

[1] https://stackexchange.com/

Table 1: Results on three retrieval tasks with two settings, using either Stack Exchange or Amazon Product Catalog databases.

		StackExchange (Any Answer)				**StackExchange (Best Answer)**				**Amazon (Future Purchase)**			
		SplitByUser		**SplitByTime**		**SplitByUser**		**SplitByTime**		**ReviewToProduct**		**ProductToProduct**	
	Method	Recall@10	Acc@100	Recall@10	Acc@100	MRR	Acc@100	MRR	Acc@100	Acc@500	Recall@1000	Acc@500	Recall@1000
	BM25-Anserini	11.45	28.33	15.79	32.64	9.64	29.49	11.68	34.79	5.71	3.51	15.09	7.48
DPR	No Expan.	36.15 ± 0.05	68.09 ± 0.14	35.46 ± 0.55	64.48 ± 0.30	20.87 ± 0.29	56.11 ± 0.09	22.87 ± 0.33	58.25 ± 0.15	6.37 ± 0.49	2.74 ± 0.20	15.54 ± 0.94	7.77 ± 0.24
	Naïve Expan.	38.76 ± 0.21	70.67 ± 0.21	38.75 ± 0.48	67.37 ± 0.45	20.03 ± 0.38	55.00 ± 0.31	21.88 ± 0.14	56.66 ± 0.33	11.04 ± 0.34	**6.10** ± 0.24	14.67 ± 1.21	7.66 ± 0.27
	Expan. w/ BM25	38.47 ± 0.34	70.37 ± 0.25	37.83 ± 0.26	66.70 ± 0.15	19.54 ± 0.18	54.08 ± 0.12	21.47 ± 0.26	56.14 ± 0.21	12.56 ± 0.36	5.89 ± 0.25	17.29 ± 0.42	8.42 ± 0.34
	DAQu (Ours)	**41.80** ± 0.27	**74.11** ± 0.24	**41.67** ± 0.39	**71.72** ± 0.33	**22.05** ± 0.24	**57.81** ± 0.80	**23.70** ± 0.18	**59.24** ± 0.46	**13.07** ± 0.19	5.97 ± 0.27	**17.86** ± 0.39	**9.15** ± 0.10
Contriever	No Expan.	42.08 ± 0.28	73.21 ± 0.15	41.93 ± 0.07	70.08 ± 0.45	25.85 ± 0.15	64.16 ± 0.34	28.37 ± 0.08	64.95 ± 0.15	8.21 ± 0.32	4.63 ± 0.20	17.80 ± 0.45	9.27 ± 0.06
	Naïve Expan.	45.25 ± 0.24	76.20 ± 0.17	44.43 ± 0.13	72.5 ± 0.18	26.01 ± 0.27	63.59 ± 0.23	28.21 ± 0.10	64.06 ± 0.36	17.23 ± 0.46	8.86 ± 0.22	17.02 ± 0.89	9.37 ± 0.53
	Expan. w/ BM25	44.69 ± 0.25	75.52 ± 0.23	44.66 ± 0.27	72.24 ± 0.39	24.71 ± 0.18	62.15 ± 0.24	27.28 ± 0.25	63.52 ± 0.55	17.71 ± 0.22	7.18 ± 0.55	17.71 ± 0.22	9.40 ± 0.21
	DAQu (Ours)	**49.74** ± 0.26	**80.27** ± 0.23	**50.28** ± 0.49	**78.06** ± 0.38	**26.47** ± 0.26	**65.16** ± 0.33	**28.82** ± 0.07	**65.47** ± 0.58	**18.75** ± 0.91	**9.86** ± 0.46	**19.87** ± 0.44	**10.42** ± 0.67

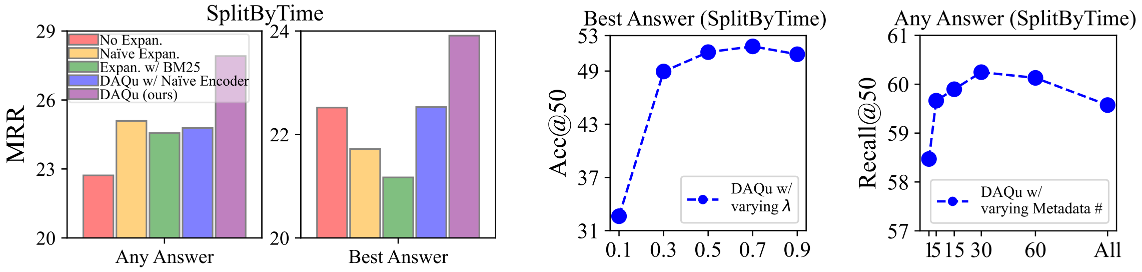


Figure 2: Analysis of the effectiveness of the set encoding strategy used in DAQu compared to a naïve encoding strategy, which simply aggregates all representations (Left), along with an investigation of our hyperparameters by varying the lambda value (Center) and the number of metadata features within each category when training DAQu (Right).

4.3 Evaluation Metrics

We report the retrieval performance with the following metrics: **1) Accuracy@K (Acc@K)** determines the fraction of queries for which the top-k results include at least one relevant document. **2) Recall@K** calculates the percentage of all relevant documents that are present within the top-k results. **3) Mean Reciprocal Rank (MRR)** computes the average of the inverse of the ranks at which the first relevant document is found across queries. **4) Mean Average Precision (MAP)** measures the mean precision score calculated after each relevant document is retrieved, across all queries.

4.4 Implementation Details

We train all retrieval models with a batch size of 16, a learning rate of 2e-5, and an AdamW (Loshchilov and Hutter, 2019). In addition, we set λ as 0.7 and randomly sample 30 features for the no-gradient metadata features in our efficient training strategy (with 3 features for gradient updates). Lastly, we report the average of three different runs.

5 Experimental Results and Analyses

We now present the overall experimental results and provide detailed analyses of our method.

Main Results We report the overall results across three different tasks with two different settings in Table 1. From this, we find that DAQu outperforms all baselines substantially, demonstrating the effectiveness of our approach that augments queries with their corresponding metadata representations (obtained from graph-based set encoding). We provide the results with additional metrics in Appendix B.1.

To be specific, for the Answer Retrieval task with Stack Exchange, while existing query expansion models achieve decent performance improvement over the no expansion baseline, our DAQu further signifies the gaps, achieving the performance improvements of 18.73% and 16.91% on SplitByUser and SplitByTime settings, respectively, against Recall@10. In addition to the Answer Retrieval task, our DAQu consistently shows superior performance on the Best Answer Retrieval task. Notably, this task is more complicated than the previous one (since the model should retrieve the post that the user mainly selects, requiring both the query-specific and the user-specific information), where query expansion baselines degrade the performance over the vanilla no expansion model. By contrast, our model is the only one that achieves performance improvement over it by large margins. Finally, the superior performance of our approach on the Future Purchase Retrieval task further confirms that it can be applicable to diverse retrieval tasks. Notably, all the aforementioned results imply that the metadata in the relational database, distributed across multiple tables, contains useful information for retrieval and that ours effectively utilizes it, unlike existing query expansion baselines that simply append the terms to the query.

Table 2: Ablation studies involving the removal or addition of each metadata category on Any Answer (SplitByTime).

	Recall		**Accuracy**	
Metadata Category	**R@20**	**Increase.**	**Acc@20**	**Increase.**
DAQu (Ours)	49.93		54.44	
w/o Comments in Q.	46.75	-6.38%	51.14	-6.06%
w/o Comments in A.	46.06	-7.74%	50.57	-7.11%
w/o Tags in Q.	49.61	-0.63%	54.29	-0.28%
No Expan.	42.22		46.39	
w/ Comments in Q.	45.24	+7.14%	49.69	+7.10%
w/ Comments in A.	47.89	+13.41%	52.31	+12.76%
w/ Tags in Q.	43.60	+3.27%	47.93	+3.31%

Table 3: Results on efficiency, based on elapsed and relative time per query, by varying the number of metadata features for category during inference on Any Answer (SplitByTime).

	Efficiency		**Effectiveness**	
# of Metadata	**Elpased**	**Relative**	**MAP**	**Acc@100**
No Expan.	0.062	1	22.94	64.15
Naïve Expan.	0.062	1.002	25.09	67.31
1 per Category	0.073	1.182	24.06	67.99
2 per Category	0.074	1.20	26.69	70.64
3 per Category	0.074	1.205	27.30	71.57
All per Category	0.075	1.218	27.53	71.98

Effectiveness of Set Encoding To see the effectiveness of the graph-based set encoding strategy when incorporating the metadata information into the query, we compare it with two types of baselines: appending their textual terms into the query or encoding them without considering the graph structure. As Figure 2 shows, simply appending the query with additional terms or taking the average of all representations in the metadata without graph structure is not as effective as ours. This demonstrates the efficacy of our two-stage (column- and query-levels) set-based metadata encoding strategy.

Analyses on Metadata Category To investigate how each category of the metadata contributes to overall performance, we conduct ablation studies by reporting the rate of performance increase when excluding or adding each category. As Figure 2 shows, each category plays a crucial role in enhancing overall performance. Furthermore, while each category does contribute to improved performance compared to the baseline without expansion, their performances are still not as high as when all categories are combined in DAQu. This implies that the information from each category is complementary to each other. Interestingly, using the 'tags' category (the information within the same table as the query) provides a small improvement, compared to using the 'comments' category from another table, which corroborates our hypothesis that it is important to use knowledge from multiple tables within the relational database.

Analyses on Hyperparameters We explore how varying the lambda value (λ) in Equation 3 (that balances the query representation with the metadata representation) impacts the overall performance in Figure 2. Specifically, when the lambda value is too low ($\lambda = 0.1$), the model fails to capture the original query's intent. Conversely, a high lambda value ($\lambda = 0.9$) leads to the model overemphasizing the original query over the metadata, thereby underutilizing the meaningful metadata representation, which degrades the performance. Thus, selecting an optimal lambda value is crucial for balancing these aspects to enhance overall performance.

We further investigate the impact of varying the number of no-gradient metadata features for each category on overall performance, when training the DAQu model. Figure 2 shows that a low count of metadata features per category results in reduced performance, indicating the importance of sufficient features for enhanced results. Yet, using all metadata features is not only inefficient but also degrades performance. Therefore, it is essential to select the appropriate number of metadata features to optimize model efficiency and effectiveness.

Analyses on Inference Efficiency We extend our investigation to the efficiency in inference, by varying the number of metadata features used for query augmentation. As Table 3 shows, although using all the metadata features during inference is effective, it requires more time compared to the model without expansion. By contrast, employing a small number of metadata features enhances efficiency while sacrificing performance. The results indicate that, at a certain point (3 features per category), there is a region where we can achieve reasonable performance alongside improved efficiency.

6 Conclusion

In this work, we presented a novel query augmentation framework, DAQu, which enhances the representation of the query with its relevant information within multiple tables over the database. To utilize the metadata features at scale with order invariance, we proposed graph-based set encoding, which hierarchically aggregates column-level and query-level information. We validated the proposed DAQu on three retrieval tasks with two settings designed with two databases, showcasing the effectiveness of our database-augmented query representation approach for information retrieval.

Limitations

While our DAQu framework effectively represents the diverse pieces of query-related metadata information (over the relational database) through a graph-structured set encoding strategy, the process of encoding and aggregating metadata representations at both the column and query levels may pose efficiency challenges in real-world applications. To address these concerns, we conducted a detailed analysis of the trade-off between the effectiveness and efficiency of DAQu in Table 3, and showcased that our approach can significantly enhance the effectiveness only with a marginal compensation of the efficiency. On the other hand, this finding still suggests that investigating more advanced methods to further increase run-time efficiency (with an approach, such as data pruning) would be a valuable direction for future research. Furthermore, the database-augmented retrieval tasks that we designed seem to be quite challenging for the retrieval models. While our DAQu generally shows significantly improved performance, there is still a large room for further improving retrieval performance.

Ethics Statement

A retrieval system can enhance the factual grounding of recent LLMs when it is integrated with them, which helps prevent the generation of plausible but incorrect answers. We believe that, following this line of directions, our DAQu can play a crucial role in diverse retrieval-augmented generation applications. Yet, it is important to note that as relational databases contain substantial amounts of knowledge, including personal information, some potential privacy concerns must be carefully managed when utilizing this information. In other words, further development of filtering strategies that tag and mask personal information across multiple tables before delivery to users or integration with LLMs would be required for real-world applications.

References

Hiteshwar Kumar Azad and Akshay Deepak. 2019. Query expansion techniques for information retrieval: A survey. *Inf. Process. Manag.*, 56(5):1698–1735.

Jagdev Bhogal, Andrew MacFarlane, and Peter W. H. Smith. 2007. A review of ontology based query expansion. *Inf. Process. Manag.*, 43(4):866–886.

Christopher Buss, Jasmin Mosavi, Mikhail Tokarev, Arash Termehchy, David Maier, and Stefan Lee. 2023. Generating data augmentation queries using large language models. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023), Vancouver, Canada, August 28 - September 1, 2023*, volume 3462 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50.

Peter Baile Chen, Yi Zhang, and Dan Roth. 2024. Is table retrieval a solved problem? join-aware multi-table retrieval. *arXiv preprint arXiv:2404.09889*.

Sukmin Cho, Jeongyeon Seo, Soyeong Jeong, and Jong C. Park. 2023. Improving zero-shot reader by reducing distractions from irrelevant documents in open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 3145–3157. Association for Computational Linguistics.

Yang Deng, Yaliang Li, Wenxuan Zhang, Bolin Ding, and Wai Lam. 2021. Toward personalized answer generation in e-commerce via multi-perspective preference modeling. *ACM Transactions on Information Systems (TOIS)*, 40:1 – 28.

Kaustubh D. Dhole and Eugene Agichtein. 2024. Genqrensemble: Zero-shot LLM ensemble prompting for generative query reformulation. In *Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part III*, volume 14610 of *Lecture Notes in Computer Science*, pages 326–335. Springer.

Yujuan Ding, Wenqi Fan, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meets llms: Towards retrieval-augmented large language models. *arXiv preprint arXiv:2405.06211*.

Massimo Esposito, Emanuele Damiano, Aniello Minutolo, Giuseppe De Pietro, and Hamido Fujita. 2020. Hybrid query expansion using lexical resources and word embeddings for sentence retrieval in question answering. *Inf. Sci.*, 514:88–105.

Zhangyin Feng, Xiaocheng Feng, Dezhi Zhao, Maojin Yang, and Bing Qin. 2024. Retrieval-generation synergy augmented large language models. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11661–11665.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. 2023. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*, abs/2312.04615.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR.

Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary C. Lipton. 2019. Amazonqa: A review-based question answering task. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4996–5002. ijcai.org.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 512–519. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *Preprint*, arXiv:2305.03653.

Nasreen Abdul Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004a. Umass at TREC 2004: Novelty and HARD. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Nasreen Abdul Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004b. Umass at TREC 2004: Novelty and HARD. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *NAACL*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2022. Augmenting document representations for dense retrieval with interpolation and perturbation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 442–452. Association for Computational Linguistics.

Soyeong Jeong, Jinheon Baek, ChaeHun Park, and Jong Park. 2021. Unsupervised document expansion for information retrieval with stochastic text generation. In *Proceedings of the Second Workshop on Scholarly Document Processing*, pages 7–17, Online. Association for Computational Linguistics.

Alistair Johnson, Tom Pollard, Lu Shen, Li-wei Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Celi, and Roger Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Sunjun Kweon, Yeonsu Kwon, Seonhee Cho, Yohan Jo, and Edward Choi. 2023. Open-wikitable : Dataset for open domain question answering with complex reasoning over table. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8285–8297. Association for Computational Linguistics.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 120–127. ACM.

Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023a. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6385–6400. Association for Computational Linguistics.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adrià de Gispert, and Gonzalo Iglesias. 2023b. LI-RAGE: late interaction retrieval augmented generation with explicit signals for open-domain table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1557–1566. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yuanhua Lv and ChengXiang Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 1895–1898. ACM.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4089–4100. Association for Computational Linguistics.

George A. Miller. 1992. WORDNET: a lexical database for english. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, USA, February 23-26, 1992*. Morgan Kaufmann.

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

Vaishali Pal, Andrew Yates, Evangelos Kanoulas, and Maarten de Rijke. 2023. Multitabqa: Generating tabular answers for multi-table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6322–6334. Association for Computational Linguistics.

Feifei Pan, Mustafa Canim, Michael R. Glass, Alfio Gliozzo, and Peter Fox. 2021. CLTR: an end-to-end, transformer-based system for cell-level table retrieval and table question answering. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, pages 202–209. Association for Computational Linguistics.

Feifei Pan, Mustafa Canim, Michael R. Glass, Alfio Gliozzo, and James A. Hendler. 2022. End-to-end table question answering via retrieval-augmented generation. *arXiv preprint arXiv:2203.16714*, abs/2203.16714.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST).

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9248–9274. 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. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9414–9423. Association for Computational Linguistics.

Zhiruo Wang, Zhengbao Jiang, Eric Nyberg, and Graham Neubig. 2022. Table retrieval may not necessitate table-specific model design. In *Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI)*, pages 36–46, Seattle, USA. Association for Computational Linguistics.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep sets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3391–3401.

Jiuling Zhang, Beixing Deng, and Xing Li. 2009. Concept based query expansion using wordnet. In *2009 International e-Conference on Advanced Science and Technology*, pages 52–55.

Wenxuan Zhang, Yang Deng, and Wai Lam. 2020. Answer ranking for product-related questions via multiple semantic relations modeling. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 569–578. ACM.

Zhi Zheng, Kai Hui, Ben He, Xianpei Han, Le Sun, and Andrew Yates. 2020. BERT-QE: contextualized query expansion for document re-ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4718–4728. Association for Computational Linguistics.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*, abs/2101.00774.

Table 4: Data statistics for each task designed with StackExchange and Amazon Product Catalog databases.

Task	Setting	Training	Valid	Test
	StackExchange			
Any Answer	SplitByUser	128,981	17,132	15,583
	SplitByTime	130,398	15,861	15,437
Best Answer	SplitByUser	43,889	6,106	5,252
	SplitByTime	42,900	6,018	6,329
	Amazon Product Catalog			
Future Purchase	ReviewToProduct	65,797	4,561	5,956
	ProductToProduct			

A Implementation Details

A.1 Datasets

In this subsection, we provide the additional details for three tasks (that we design) based on the StackExchange and Amazon Product Catalog datasets. We first report the detailed statistics of the overall datasets in Table 4. In addition to this, in Table 6, we present more fine-grained statistics of each category (column) of the metadata, used for each query. Notably, in this table, we breakdown the metadata features into two categories: 'total query' (that includes all the queries in the task) and 'non-empty query' (that includes queries that have at least one item for each specific metadata category).

Stack Exchange Recall that, for this database, we design two tasks: **1) Answer Retrieval (Any Answer)** and **2) Best Answer Retrieval (Best Answer)**. In this paragraph, we describe which specific metadata categories that we use for query augmentation. At first, for the Answer Retrieval task, we utilize metadata from the post and comment tables. Specifically, we focus on the tags associated with the current question post and the comments on both the current question and the answer posts. For the Best Answer Retrieval task, we utilize metadata from the post, comment, vote, and user tables. The reason why we utilize more categories for this task is because this task is closely related to the personalized retrieval task (for the user who issues the question post); therefore, we focus on constructing the user-specific metadata. Specifically, we use the total comments made by the user, the 'aboutme' information of the user, written question and answer posts, and the voted answer posts by the user. Additionally, we include tags from both the current question post and previously asked question posts. For both tasks, we split the queries with their corresponding metadata into training, validation, and test sets, using a corpus of 3,281,834 documents that contain all posts, according to two different settings. In the SplitByUser setting, we randomly sample users in an 8:1:1 ratio from those who have posted questions with answers provided by others. On the other hand, for the SplitByTime setting, we split the datasets based on the creation timestamp of the question posts. Specifically, we create a training set with question posts written before 2019-01-01, a validation set with posts written after 2019-01-01 but before 2020-01-01, and a test set with posts written after 2020-01-01.

Amazon Product Catalog For this database, we design the **3) Future Purchase Retrieval (Future Purchase)** task, where we utilize all the user, product, and review tables. Furthermore, we consider the book reviews written from 2013-01-01 to 2016-01-01 (due to the size of the entire corpus), constructing a document corpus using each product's description, Specifically, we use reviews written in 2013 for the training set, reviews in 2014 for the validation set, and reviews in 2015 for the test set. We then group the reviews written by each customer and randomly sample the customers (since the data before sampling is still very large), selecting 5,000 for the training set, 500 for the validation set, and 500 for the test set. Among two different settings for this task, in the ReviewToProduct setting, each review text (input) is paired with future products (target) that the customer will purchase. For this setting, we incorporate metadata from the previous review text from the review table, and the category, title, and description of both the current and previous products from the product table. In the ProductToProduct setting, we pair the product description of the current review with future products that the customer will buy. We utilize metadata from both the current and previous review texts from the user's review table, along with the category and title of both current and previous products, and the description of the previous products.

A.2 Models

For DPR (Karpukhin et al., 2020), we follow the implementation by Thakur et al. (2021). For Contriever (Izacard et al., 2022), we further train it from its available checkpoint, while using the same architecture as DPR. For a fair comparison, we fix the number of epochs across the same retrieval models for each task and report the average of the three different runs for every model. We use A100 GPU clusters for conducting experiments.

B Experimental Results

B.1 Additional Results with Different Metrics

In addition to our main results in Table 1, we provide the results with other retrieval metrics in Table B.1. From this, similar to the results in Table 1, we also observe that our DAQu shows remarkable performance improvements in diverse scenarios.

B.2 Case Study

We conduct a case study to qualitatively compare the effectiveness of our DAQu against the baseline query augmentation methods, provided in Table 7. The first example from the Any Answer retrieval task with the SplitByTime setting presents retrieval results for a user query: selecting optimal activation and loss functions when training an autoencoder on the MNIST dataset. Notably, the challenge here is several important keywords with query-relevant information, such as BCE and MSE, are missing from the original user query. While the baseline expansion models can include such keywords, which can lead to a higher rank of the relevant document (Naïve Expansion), Expansion with BM25 results in a lower rank than even No Expansion, due to the exclusion of another essential term, ’Keras’. In contrast, our DAQu achieves the highest rank among all baselines, indicating that our method effectively augments all essential information with the metadata representation, by utilizing diverse useful information sources in a relational database. Similarly, for the Best Answer retrieval task with the SplitByTime setting, given a query such as when normalization or standardization is appropriate, the best answer post explains such cases in terms of ‘transformation methods.’ Here, our DAQu, which can incorporate the relevant term ‘log transformation’ from the metadata into the query representation, achieves the highest rank. Finally, for the Future Product retrieval task, a user purchased the book ‘Kindergarten-Grade 3’ for their children. In addition, this user’s metadata includes information on several previous purchases tagged ‘Children’s Books.’ In this example, while the No Expansion baseline effectively retrieves the future product with a higher rank, Naïve Expansion and Expansion with BM25 do not perform well, suggesting that augmenting metadata with text level adds noise to the retrieval process. Meanwhile, our proposed method effectively exploits only the useful information on the latent space, achieving the highest rank among all models.

Table 5: Additional Results on three retrieval tasks with two settings on Stack Exchange and Amazon Product Catalog databases.

		StackExchange (Any Answer)				**StackExchange (Best Answer)**				**Amazon (Future Purchase)**			
		SplitByUser		**SplitByTime**		**SplitByUser**		**SplitByTime**		**ReviewToProduct**		**ProductToProduct**	
	Method	MAP	MRR	MAP	MRR	Acc@10	Acc@50	Acc@10	Acc@50	Acc@1000	Recall@500	Acc@1000	Recall@500
DPR	No Expan.	23.56 ± 0.03	27.86± 0.08	22.72 ± 0.22	25.22 ± 0.24	32.75 ± 0.23	48.63 ± 0.20	35.11 ± 0.60	50.96 ± 0.55	9.23 ± 0.19	1.78 ± 0.27	19.73 ± 0.85	5.98 ± 0.44
	Naïve Expan.	25.63 ± 0.03	30.15 ± 0.07	25.16 ± 0.11	27.85 ± 0.14	31.44 ± 0.47	47.13 ± 0.41	33.81 ± 0.33	49.27 ± 0.27	16.10 ± 0.92	**4.55** ± 0.24	20.74 ± 1.13	5.54 ± 0.37
	Expan. w/ BM25	25.31 ± 0.04	29.79 ± 0.05	24.55 ± 0.05	27.19 ± 0.09	30.98 ± 0.07	46.60 ± 0.31	33.27 ± 0.15	48.72 ± 0.17	17.77 ± 0.36	4.13 ± 0.21	22.65 ± 0.74	6.50 ± 0.13
	DAQu (Ours)	**27.96** ± 0.23	**32.86** ± 0.10	**27.58** ± 0.31	**30.37** ± 0.35	**33.99** ± 0.25	**50.05** ± 0.33	**36.14** ± 0.42	**52.20** ± 0.47	**18.01** ± 0.29	4.23 ± 0.21	**22.68** ± 1.08	**7.06** ± 0.15
Contriever	No Expan.	28.46 ± 0.23	33.23 ± 0.19	28.38 ± 0.28	31.22 ± 0.31	39.71 ± 0.42	56.13 ± 0.33	42.07 ± 0.43	57.90 ± 0.20	12.62 ± 0.73	3.14 ± 0.26	21.76 ± 0.37	7.65 ± 0.19
	Naïve Expan.	31.06 ± 0.16	36.12 ± 0.12	30.12 ± 0.08	33.14 ± 0.08	39.28 ± 0.35	56.04 ± 0.43	41.32 ± 0.15	57.33 ± 0.53	22.65 ± 0.67	7.07 ± 0.14	23.60 ± 0.88	7.14 ± 0.36
	Expan. w/ BM25	30.82 ± 0.19	35.76 ± 0.22	30.30 ± 0.32	33.24 ± 0.35	38.09 ± 0.50	54.56 ± 0.25	40.79 ± 0.45	56.42 ± 0.41	22.62 ± 0.22	5.42 ± 0.44	22.62 ± 0.22	7.44 ± 0.04
	DAQu (Ours)	**35.00** ± 0.33	**40.55** ± 0.41	**34.96** ± 0.53	**38.07** ± 0.57	**40.50** ± 0.16	**57.59** ± 0.58	**42.53** ± 0.06	**58.48** ± 0.51	**25.65** ± 0.44	**7.10** ± 0.29	**25.36** ± 0.50	**8.31** ± 0.23

Table 6: Distribution of the metadata features per query for each metadata category for three retrieval tasks.

		Total Query			**Non Empty Query**		
Setting	**Metadata Category**	**Training**	**Valid**	**Test**	**Training**	**Valid**	**Test**
				StackExchange - Any Answer			
SplitByUser	comments_in_question	1.96	1.95	1.94	3.35	3.37	3.31
	comments_in_answers	2.31	2.45	2.31	3.96	4.14	3.99
	tags	3.00	3.04	3.01	3.00	3.04	3.01
SplitByTime	comments_in_question	2.03	1.69	1.63	3.38	3.19	3.26
	comments_in_answers	2.43	1.89	2.08	4.09	3.46	3.71
	tags	2.97	3.06	3.23	2.97	3.06	3.23
				StackExchange - Best Answer			
SplitByUser	question_posts	14.52	22.15	12.42	18.18	27.07	15.77
	answer_posts	19.77	24.25	13.47	44.79	55.18	30.74
	accepted_answers	7.41	13.41	6.25	10.91	18.68	9.41
	comments	81.28	122.02	84.92	92.86	137.92	97.46
	aboutme	0.33	0.31	0.33	1.00	1.00	1.00
	current_tags	3.06	2.99	3.08	3.06	2.99	3.08
	previous_tags	48.36	66.99	41.59	48.36	66.99	41.59
SplitByTime	question_posts	6.52	7.04	9.96	10.46	11.25	14.94
	answer_posts	7.82	9.35	11.15	27.47	38.98	42.83
	accepted_answers	3.82	3.67	5.36	7.29	7.21	9.77
	comments	31.09	38.59	49.44	54.32	67.36	81.55
	aboutme	0.34	0.29	0.28	1	1	1
	current_tags	3.02	3.10	3.25	3.02	3.10	3.25
	previous_tags	19.52	21.71	32.33	31.31	34.70	48.52
				Amazon Product Catalog			
ReviewToProduct	previous_review_text	8.22	6.97	15.05	11.22	8.94	17.52
	current_product_category	2.90	2.91	2.86	2.99	3.00	2.99
	current_product_title	1.00	1.00	1.00	1.00	1.00	1.00
	current_product_description	1.00	1.00	1.00	1.00	1.00	1.00
	previous_product_category	23.96	20.34	44.16	33.01	26.39	52.68
	previous_product_category	8.22	6.97	15.05	11.22	8.94	17.52
	previous_product_description	8.22	6.97	15.05	11.22	8.94	17.52
ProductToProduct	previous_review_text	8.22	6.97	15.05	11.22	8.94	17.52
	current_product_category	2.90	2.91	2.86	2.99	3.00	2.99
	current_product_title	1.00	1.00	1.00	1.00	1.00	1.00
	current_product_description	1.00	1.00	1.00	1.00	1.00	1.00
	previous_product_category	23.96	20.34	44.16	33.01	26.39	52.68
	previous_product_category	8.22	6.97	15.05	11.22	8.94	17.52
	previous_product_description	8.22	6.97	15.05	11.22	8.94	17.52

Table 7: Case study on three retrieval tasks. In response to the query from the user, notable terms in the Metadata and Answer Post are highlighted in red, which are not in the query but exist only in the metadata and answer posts. Additionally, among those notable terms, some terms that are not covered by the query expansion approach are further highlighted in **bold**.

StackExchange-Any Answer w/ SplitByTime	
Query	**[Title]** Choosing activation and loss functions in autoencoder **[Text]** I am following this keras tutorial to create an autoencoder using the MNIST dataset. Here is the tutorial: <URL>. However, I am confused with the choice of activation and loss for the simple one-layer autoencoder (which is the first example in the link). Is there a specific reason sigmoid activation was used for the decoder part as opposed to something such as relu? I am trying to understand whether this is a choice I can play around with, or if it should indeed be sigmoid, and if so why? Similarily, I understand the loss is taken by comparing each of the original and predicted digits on a pixel-by-pixel level, but I am unsure why the loss is binary crossentropy as opposed to something like mean squared error. I would love clarification on this to help me move forward! Thank you!
MetaData	**[comments in answers by pid]**: ["I wrote about it here, but it was ages ago so I cannot find it now; BCE's properties as a function means it's not the best choice for image data, even in greyscale. Unlike MSE, it is asymmetrically biased against overconfidence, so it systematically underestimates the values, needlessly dimming the output intensities. And, as this question shows, causes unnecessary confusion on top.", "Hmm. I think you may be correct in general, but for this particular use case (an autoencoder), it's been empirically and mathematically shown that training on the BCE and MSE objective both yield the same optimal reconstruction function: <URL> — but that's just a minor detail.", "I cannot load the pdf for some reason, but I'm not surprised - the minima of both losses are the same if your goal is to autoencode a 1:1 match of intensities. It's just not always an optimal loss if your goal is to have a nice-looking image; e.g. MNIST would probably look best with most pixels being either 1 or 0 (in/not in the set of pixels for the character, basically learning a topology)."], **[tags by pid]**: ['neural-networks', 'loss-functions', '**keras**', 'autoencoders']
Answer Post	You are correct that MSE is often used as a loss in these situations. However, the **Keras** tutorial (and actually many guides that work with MNIST datasets) normalizes all image inputs to the range [0, 1]. This occurs on the following two lines: `x_train = x_train.astype(float32) / 255`, `x_test = x_test.astype(float32) / 255`. Note: as grayscale images, each pixel takes on an intensity between 0 and 255 inclusive. Therefore, BCE loss is an appropriate function to use in this case. Similarly, a sigmoid activation, which squishes the inputs to values between 0 and 1, is also appropriate. You'll notice that under these conditions, when the decoded image is "close" to the encoded image, BCE loss will be small. I found more information about this <URL>.
Retrieval Rank	No Expan. : 26 Naïve Expan. : 15 Expan. w/ BM25 : 38 DAQu (Ours) : 6
StackExchange-Best Answer w/ SplitByTime	
Query	**[Title]** When to Normalization and Standardization? **[Text]** I see pro-processing with Normalization, which aligns data between 0 and 1, and standardization makes zero mean and unit variance. And multiple standardization techniques follow on.. Any clear definition at what cases what should be used? Thanks in Advance!!
MetaData	**[comments]**: ['hi @onestop, is it ok to take **log transformation** only to skewed columns?'] **[current tags]**:['normalization', 'feature-scaling']
Answer Post	In unsupervised learning, the scaling of the features has a great influence on the result. If a feature has a variance that is many times greater, it can dominate the target function of the algorithm. Therefore, it is of great importance to scale the input data in a way that their variability matches or at least does not contradict the semantics. There are several **transformation methods** to put the features into a comparable form. These use different forms of normalization or standardization according to their context. (...)
Retrieval Rank	No Expan. : 244 Naïve Expan. : 178 Expan. w/ BM25 : 347 DAQu (Ours) : 105
Amazon-Future Purchase w/ ProductToProduct	
Query	Kindergarten-Grade 3. Fox has composed a simple refrain to celebrate human connections in this lovely picture book. "Little one, whoever you are," she explains, there are children all over the world who may look different, live in different homes and different climates, go to different schools, and speak in different tongues but all children love, smile, laugh, and cry. Their joys, pain, and blood are the same, "whoever they are, wherever they are, all over the world." Staub's oil paintings complement the simple text. She uses bright matte colors for the landscapes and portraits, placing them in gold borders, set with jewels and molded from plaster and wood. These frames enclose the single- and double-page images and echo the rhythm of the written phrases. Within the covers of the book, the artist has created an art gallery that represents in color, shape, and texture, the full range of human experience.
MetaData	**[previous product description]**:["Betsy Snyder's first board book as an author-illustrator, Haiku Baby follows a tiny bluebird, the book's would-be protagonist, as it visits its various animal companions–from an elephant that shades the bird with a parasol to a fox in a meadow and a whale in the ocean. The little bird's story is told primarily in pictures, and through the book's six haiku: rain, flower, sun, leaf, snow, and–of course, it would not be a board book without–the moon, making it ideal for the bedtime line-up. Adorable cookie-cut illustrations work nicely with the haiku form to give the book a whimsical, yet serene, feel. And the haiku are light and fun without being too cutesy. Index tabs on the right margin, with pictures that tie to each of the poems (leaf, raindrop, snowflake, etc.), create a unique look, and make it easy for toddlers to flip through the pages on their own without having them stick together like they can with other board books. Snyder excels at visual storytelling and short forms, possibly a talent she honed as a designer/illustrator in the kids' greeting card business. In the world of board books, this slender little volume really stands out"] **[previous product category]**:["Books", "**Children's Books**", "Early Learning"] **[previous review text]**:["My baby loves this book. It has been mouthed, pulled, and thrown many times and still looks new. No tears or running on the pages. No words inside, but has the song on the back incase one does not know it. Can easily make your own story up. My sister washed her book, which you should not do, and it got wrinkled and looks worn down. It did not tear or come apart though", 'Nice little book. Has all the seasons and some weather.']
Future Product	**[Title]** Ten Little Fingers and Ten Little Toes **[Text]** "There was one little baby who was born far away. And another who was born on the very next day. And both of these babies, as everyone knows, had ten little fingers and ten little toes." So opens this nearly perfect picture book. Fox's simple text lists a variety of pairs of babies, all with the refrain listing the requisite number of digits, and finally ending with the narrator's baby, who is 11truly divine" and has fingers, toes, 11and three little kisses/on the tip of its nose." Oxenbury's signature multicultural babies people the pages, gathering together and increasing by twos as each pair is introduced. They are distinctive in dress and personality and appear on primarily white backgrounds. The single misstep appears in the picture of the baby who was "born on the ice." **The child**, who looks to be from Northern Asia or perhaps an Inuit, stands next to a penguin. However, this minor jarring placement does not detract enough from the otherwise ideal marriage of text and artwork to prevent the book from being a first purchase. Whether shared one-on-one or in storytimes, where the large trim size and big, clear images will carry perfectly, this selection is sure to be a hit."
Retrieval Rank	No Expan. : 29 Naïve Expan. : 162 Expan. w/ BM25 : 765 DAQu (Ours) : 27