Multi-hop Database Reasoning with Virtual Knowledge Graph

Juhee Son[1], **Yeon Seonwoo**[1], **Seunghyun Yoon**[2], **James Thorne**[1], **Alice Oh**[1]
[1]KAIST, [2]Adobe
{sjh5665, yeon.seonwoo, thorne}@kaist.ac.kr,
syoon@adobe.com, alice.oh@kaist.edu

Abstract

Application of LLM to database queries on natural language sentences has demonstrated impressive results in both single and multi-hop scenarios. In the existing methodologies, the requirement to re-encode query vectors at each stage for processing multi-hop queries presents a significant bottleneck to the inference speed. This paper proposes **VKGFR** (**V**irtual **K**nowledge **G**raph based **F**act **R**etriever) that leverages large language models to extract representations corresponding to a sentence's knowledge graph, significantly enhancing inference speed for multi-hop reasoning without performance loss. Given that both the queries and natural language database sentences can be structured as a knowledge graph, we suggest extracting a Virtual Knowledge Graph (VKG) representation from sentences with LLM. Over the pre-constructed VKG, our VKGFR conducts retrieval with a tiny model structure, showing performance improvements with higher computational efficiency. We evaluate VKGFR on the WikiNLDB and MetaQA dataset, designed for multi-hop database reasoning over text. The results indicate 13x faster inference speed on the WikiNLDB dataset without performance loss.

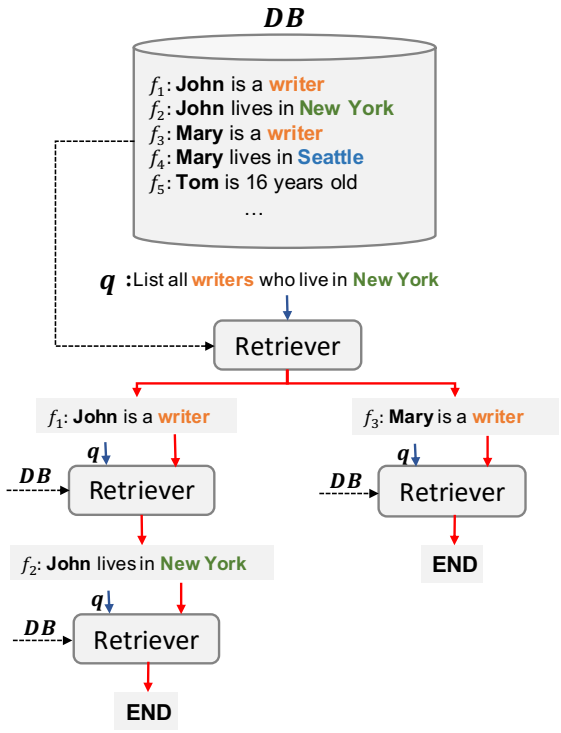


Figure 1: Illustration of the multi-hop reasoning procedure for answering the database query. The retriever searches related facts in the database per each reasoning step. As the database sizes increases, the number of facts used for multi-hop reasoning increases, so the number of retriever's operation gets higher.

1 Introduction

If open-domain question-answering models could accurately reason with large-scale facts in databases, it would make it feasible to substitute or augment existing database management systems with NLP technology (Thorne et al., 2021b). Several benchmarks have been proposed (Weston et al., 2016a; Dua et al., 2019; Thorne et al., 2021a), which range in size and complexity and require systems to conduct discrete reasoning (incorporating numerical operations like counting and argmax) by collating multiple facts within the database. To facilitate the reasoning at the scale of databases, NLP systems are required to access numerous sets of relevant sentences, often in combination with multi-hop retrieval (Figure 1).

For open-domain NLP systems to reason over only the relevant subset of facts from a corpus, a two-stage architecture of retrieval and reasoning is typically used (Petroni et al., 2021). However, challenges in database-style reasoning require additional complexity with retrieving non-redundant sets of tens or hundreds of facts. For the WikiNLDB benchmark, Thorne et al. (2021a) proposed an incremental retrieval architecture called

Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024), pages 1–11
August 15, 2024 ©2024 Association for Computational Linguistics

SSG (Support Set Generator) built on sentence-BERT (Reimers and Gurevych, 2019). The inference speed of the SSG is not scalable because it encodes query vectors for each reasoning step in multi-hop retrieval through the transformer (Vaswani et al., 2017), causing computation inefficiency. As the size of the database increases, retrieval slows down significantly. For example, with 25 facts in the database, SSG processes 21.51 queries per second, but as the number of facts increases to 1000, SSG only processes 1.46 queries per second.

In this paper, we propose **VKGFR** (**V**irtual **K**nowledge **G**raph based **F**act **R**etriever) which significantly improves the inference speed and retrieval performance. Since the WikiNLDB dataset can be represented in the knowledge graph structure, we suggest constructing a virtual knowledge graph (Wang et al., 2017, VKG) for fact retrieval with LLM. The VKG embedding provides compressed vectorized representations of facts and queries and can be pre-indexed, enabling efficient and accurate multi-hop retrieval. Recent works have used VKG to predict target entities from knowledge bases (Dhingra et al., 2020; Sun et al., 2021) or retrieve facts to claim verification (de Jong et al., 2021). However, the number of reasoning steps and hop lengths are predetermined for these specific tasks, making it challenging to adapt them directly to database reasoning. Applying the VKG to the database reasoning task is non-trivial because it requires various hops of reasoning, and the candidates per each reasoning step are not known a priori, and our work is the first to employ VKG for database reasoning.

We evaluate VKGFR on WikiNLDB, a database reasoning task consisting of various sizes of database facts and corresponding queries, and MetaQA (Zhang et al., 2018), a conventional multi-hop QA dataset over the knowledge base. VKGFR performs best compared with several other VKG-based models and multi-hop dense retrieval models (Xiong et al., 2021; Lee et al., 2021) on both datasets. Furthermore, VKGFR shows at least 4.7 times faster inference time than SSG in all database sizes of WikiNLDB (Figure 6). We conduct an ablation study with different types of entity embedding and model structure, and our approach shows the best performance. Our main contribution is to propose a significantly more efficient and accurate VKG-based retriever enabling natural language database reasoning.

Facts
- John is a writer who lives in Seoul, 35 years old
- Marry is 18 year old girl graduated from Boston school
- James is a 40 years old lawyer graduated from Harvard law school

Queries
- **Argmax**: Who is the oldest person?
- **Set**: List all writers lives in New York.
- **Count**: How many people live in Seoul?
- **Bool**: Did James graduate from Harvard?

Figure 2: Examples of facts included in natural language databases and database queries. The highlighted texts are entities important for reasoning on the database.

2 Background

2.1 Natural Language Databases (NLDBs)

Natural language databases (Thorne et al., 2021a,b) model large collections of facts stored in plain text as the storage media for database reasoning. In contrast to open-domain question answering, database reasoning requires making inferences over large sets of facts related to one query. Conventional open-domain question-answering methods need to encode all relevant facts, possibly in the thousands, perform discrete reasoning to get the most related facts, and then decode a sequence of tokens representing the answer to the query. Previous works have studied small synthetic settings (Weston et al., 2016b) or reasoning over a single passage (Dua et al., 2019).

Conventional databases store facts in structured forms with labeled columns and are queried with formal languages such as SQL. Much work in NLP has studied the parsing of user queries into structured representations or exposing the database through a natural language interface (Androutsopoulos et al., 1995; Zhong et al., 2017). However, in NLDBs, because *both* the stored text and queries are natural languages, NLDBs are not restricted by any predefined database schema allowing the addition of new topics without defining tables or columns, reducing maintenance overheads.

NLDBs are studied using the WikiNLDB dataset (Thorne et al., 2021a), which contains databases varying in size (from 25 to 1000 facts) and question-answer pairs. An example is provided in Figure 2. In WikiNLDB, four different types of queries require different reasoning processes (specifically, counting, min/max, argmin/argmax, and set-based

answers) and over single entities and short multi-hop chains (referred to as *joins*).

2.2 Virtual Knowledge Graphs (VKGs)

Our proposed solution is to perform retrieval by modeling the set of facts as a VKG: a knowledge graph representation where pairs of entities and the relation between them is embedded: $(m_{e_1}, m_{e_2}, \vec{r}_{(e_1,e_2)})$. VKG representations have been used for retrieval in QA, but their usage and constructions vary by application. **DrKIT** (Dhingra et al., 2020) and **TOME** (de Jong et al., 2021) used the entity representation as a memory bank for fixed-length multi-hop retrieval. **OPQL** (Sun et al., 2021) constructs a key-value memory with VKG for the fixed length of multi-hop retrieval and multi-hop slot filling task. The *key* in OPQL is the concatenation of the target entity embedding and the relation vector, and the remaining entity embedding becomes the *value* of the memory. **VKGDR** (Seonwoo et al., 2022) uses VKG for zero-shot domain-specific retrieval and calculates the relevance score of queries and documents by multiplying the relation vectors. In contrast to previous work in VKG-based retrieval, which uses a subset of the VKG for a fixed number of hops, we use the whole VKG representation to perform variable-length multi-hop retrieval.

3 Methods

Our multi-hop fact retriever **VKGFR** comprises two key steps: first, facts and queries are embedded into VKGs (Section 3.1, Figure 3); second, multi-hop retrieval is performed over the embedded VKG (Section 3.2, Figure 4). In contrast to SSG (Thorne et al., 2021a), the embeddings of facts are immutable and can be pre-indexed, yielding faster retrieval. For inference, VKG embeddings of new facts or queries can be embedded on demand.

3.1 Building the VKG

Entity Encoder We extract the entity spans from the text with a predefined entity vocabulary built over the Wikipedia entities. All possible pairs of extracted entities become part of the VKG. We use a pre-trained language model to compute the contextualized embeddings of those entities. We experiment with various models (Karpukhin et al., 2020; de Jong et al., 2022; Devlin et al., 2019) and use the best-performing model, *DensePhrase*.

Relation Encoder The relation encoder computes a relation vector between a pair of entities (Seonwoo et al., 2022; Sun et al., 2021; Baldini Soares et al., 2019). Consistent with previous approaches, the input to the relation encoder is a sentence with two entities masked and a special relation token inserted behind each masked token (e.g. "`[ENT] [R1]` is a `[ENT] [R2]` who lives in L.A."). The masking makes the model learn the relation representation based on the context of the entities rather than the textual representation itself. For the two relation tokens, the relation vector is computed by concatenation and linear projection:

$$\vec{r}_{e_1,e_2} = W^{\intercal}[h_{r_1}; h_{r_2}] \quad (1)$$

Hyper parameter described on section 8

The relation encoder is trained with supervision that relations containing the same entity pairs are located in a similar vector space. The relation vectors that consist of the same entity pairs are regarded as positive samples. Below is the cross-entropy training loss for the relation encoder:

$$L_{rel\_enc} = \text{CE}(\sigma(\vec{r}^{\intercal}_{e_1,e_2}\vec{r}_{e_i,e_j}), \mathbb{I}_{(e_1=e_i,e_2=e_j)}) \quad (2)$$

Following Sun et al. (2021), we pre-train the relation encoder with Wikidata and fine-tune it on the respective target datasets (WikiNLDB, MetaQA).

VKG for WikiNLDB As described above, we build VKG embeddings for the facts and queries in WikiNLDB. For the facts, we extract the entity span

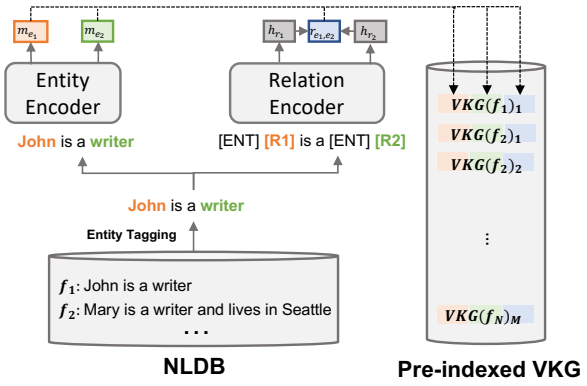


Figure 3: Illustration of our VKG construction method. Each encoder independently builds the vectorized representation of facts. The concatenation of the entity and relation embeddings becomes our VKG representations of the text. For the f_2, there are two entity pairs (Mary-writer, Mary-Seattle) so the corresponding VKG representation is indexed as two different triplets $(VKG(f_2)_1, VKG(f_2)_2)$.

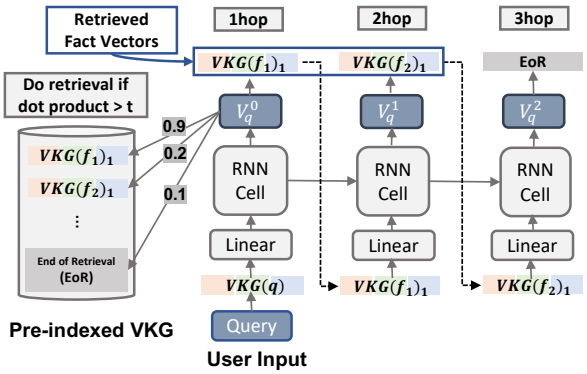


Figure 4: Illustration of VKGFR's multi-hop inference procedure. The query is encoded with an RNN, and the retrieved fact (f_1) becomes the input for the next hop. This multi-hop reasoning process is repeated until the model predicts the special END vector, indicating sufficient information was returned.

(e_i, e_j) based on the Wikipedia entity vocabulary $\mathcal{E}$ and embed them with pretrained models (m_{e_i}, m_{e_j}) and calculate a relation vector $r_{(e_i,e_j)}$ for each pair. As in Figure 2, facts can contain many entity pairs, so there can be multiple mention-relation-mention triplets of the same fact. The VKG representations for fact f that have n entity pairs are denoted as follows:

$$vkg(f) = \{[m_{e_i}; m_{e_j}; r_{e_i,e_j}]_k\}_{k=1}^{n}, \quad \forall i\ e_i \in \mathcal{E} \tag{3}$$

For queries, we mask the entity and add the special relation token at the end of the sentence to compute the relation vector (e.g. "How many people study at `[ENT][R1]`? `[ENT][R2]`"). If there are multiple entities in the query, we average the VKG representation per each entity and take the average because the query is a single unit used for comparison, so it is more beneficial to include all entity pair relations in the query. The following is VKG representation for query q that has n entities:

$$vkg(q) = \frac{1}{n}\sum_{k=1}^{n}[m_{e_i}; m_{e_j}; r_{e_i,e_j}]_k, \ \forall i\ e_i \in \mathcal{E} \tag{4}$$

3.2 Multi-hop Retriever

Figure 4 depicts the comprehensive inference mechanism of VKGFR. VKGFR retrieves relevant facts by searching over a pre-indexed fact VKG with the given query VKG. For each retrieval step, VKGFR applies a linear layer to project the fact (W_f) and query (W_q) VKG embeddings. Then, to encode the multi-hop aspect of retrieval, we apply an RNN layer to transform the vector (Equation 5), considering the retrieval history.

$$V_q^0, h^0 = RNN(W_q^\intercal vkg(q), 0) \in \mathbb{R}^D \tag{5}$$

Using the query vector, the fact that the relation probability is over the threshold (τ) is returned ($vkg(f_t)$ = *retrieve*(V_q^t,τ)). For each retrieval hop, the retrieved fact vector becomes the input of the next step (Equation 6)

$$V_q^{t+1}, h^{t+1} = RNN(W_f^\intercal vkg(f_t), h^t) \in \mathbb{R}^D \tag{6}$$

The retrieved facts are further processed by VKGFR, repeating this retrieval step until a special End-of-retrieval (EoR) vector is retrieved.

Training We optimize the cross entropy loss between the inner products of fact, query vectors, and the ground truth (gt) label that is 1 if the fact is correct for the query and 0 otherwise.

$$L_{retreiver} = \text{CE}(\sigma(V_{f_i}^\intercal V_q^t), \mathbb{I}_{f_i \in gt(q)}) \tag{7}$$

Retrieval The relevance probability between the query and fact is estimated by computing the inner product of the query and fact vectors. If this probability exceeds a hyper-parameterized threshold $\tau = 0.5$, the fact is retrieved. To model multi-set multi-hop retrieval for WikiNLDB, the retrieval process branches if more than one fact is retrieved. Each branch is independently decoded until EoR is predicted.

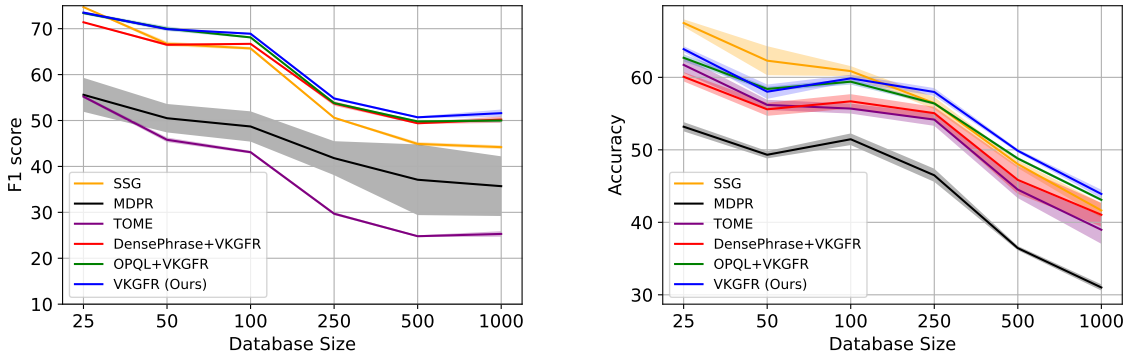
4 Experiments on WikiNLDB

4.1 Experimental Setup

Data The WikiNLDB dataset consists of databases between 25 and 1000 facts. The size of the database defines the upper bound of the number of candidates for retrieval. Following the original paper (Thorne et al., 2021a), we use the training data from the database size 25 and train a single model which was tested for all sizes.

VKG Embedding For the entity encoder, we use the publicly available BERT-base size DensePhrase checkpoint[1]. For the relation encoder, we pretrained the BERT-large with the Wikidata and fine-tuned this on WikiNLDB.

[1] https://github.com/princeton-nlp/DensePhrases



(a) Retriever performance

(b) Reader performance

Figure 5: Retriever and reader performance of models on all database sizes

Retriever VKGFR is trained with the pre-indexed VKGs. We use one RNN layer for multi-hop retrieval, chosen empirically. We sample hard negative facts that share the same entity or same relation with the ground truth facts for training. We set the sample ratio as 1:10 and the threshold as 0.5 based on empirical performance on the validation set.

Full Pipeline (With Reader) The contributions in this paper focus on the retrieval side of a two-part architecture. For completeness, we experiment with the reader component. We use the pre-existing NeuralSPJ model from Thorne et al. (2021a). This model is an encoder-decoder transformer based on the T5 architecture that generates a machine-readable version of a natural language fact given a query if the fact is relevant or no output otherwise. Following previous approach, we trained this model using the gold passages from WikiNLDB and sampled false-positive facts from our retriever for resilience. To train the model to predict no output for false-positive retrieved facts, we sample false positives from our retrieved facts.

For evaluation, we report precision, recall, and F1 score for the retriever and answer exact-match from the reader. To evaluate variance, we run each experiment with three seeds and average the results. Appendix 8 describes the hyperparameters.

4.2 Retrieval Baselines

SSG (Thorne et al., 2021a) is a SentenceBERT-based multi-hop retriever, using an inner-product-based search mechanism with branching for multi-hop retrieval. **TOME** (de Jong et al., 2022) uses predefined mention encoding for multi-hop retrieval. This shows the best performance on the fact verification task. We fine-tune TOME for WikiNLDBs. **MDPR** (Xiong et al., 2021) uses dense representation for multi-hop retrieval, iteratively encoding the questions using the question encoder. To apply MDPR to the WikiNLDB dataset, the number of candidates retrieved for every reasoning step needs to be set, and we set the number as the maximum reasoning steps of the NLDB training data (Asai et al., 2020). **DensePhrase** (Lee et al., 2021) is the text retrieval model we use for entity embedding, and we experiment with only the DensePhrase embedding on our model to figure out the effect of *our* relation embedding. **OPQL** (Sun et al., 2021) memory uses VKG for multi-hop reasoning, but their VKG representations consist of only the relation vector and target entity embedding, so we compare our VKG building method to the OPQL memory. To enable variable lengths of multi-hop reasoning on DensePhrase and OPQL, we add VKGFR over the pre-indexed DensePhrase and OPQL embedding.

4.3 Ablation Study

To verify our VKG encoding method, we conduct an ablation study with the following types of entity embedding: 1) DensePhrase (Lee et al., 2021, DP) records the dense representation of passages, which can be a single entity. 2) Mention Encoder (de Jong et al., 2022, ME) encodes dense vector representations of every entity mention in a text, which is built on the transformer architecture, and the entity span is projected to the fixed-sized vector space. 3) The average value of BERT (Devlin et al., 2019) hidden embedding between the entity span used as an entity representation. We build VKG triplets based on different embedding models and trained VKGFR over those representations with the

same hyperparameter. 4) We compare the VKGFR model structure between the *RNN* and *Linear* layer.

5 Results on WikiNLDB

5.1 Overall Results on Whole Databases

Figure 5 describes the overall performance of the retriever and reader for all sizes.

Retriever Performance For retrieval performance, VKGFR shows the best or comparable F1 score on all database sizes. SSG performs best on the smallest database size (25 facts), but as the database size increases, the performance drastically drops. VKGFR is consistently better than DensePhrase and OPQL, which means that our VKG-building methods are effective in improving retrieval. Compared to other retrieval baselines, MDPR shows low performance with high variance, indicating that the fixed number of candidates had a negative impact on performance. TOME shows the lowest performance, implying that the mention encoding strategy of TOME is not effective on this dataset.

Reader Performance VKGFR shows the best performance on large database sizes (>100), but the SSG is better on smaller database sizes. DensePhrase, TOME, and OPQL showed consistently lower performance than VKGFR on all database sizes. MDPR showed much lower performance for the reader even though it showed a better retrieval score than the TOME, caused by noise from the fixed number of retrieval candidates.

5.2 Results for Different Types of Queries

WikiNLDB consists of four different types of queries: min/max, set, count, and boolean. We analyze results from the models with different types of queries on the largest database size (1000 facts).

Retriever Performance We report retrieval results for all models in Table 1. VKGFR shows the highest F1 score on most query types, min/max, set, and count. In comparison to the SSG, VKGFR showed better precision which leads better F1 score but the recall score of SSG is higher than VKGFR. TOME and MDPR showed the lowest precision but comparable recall scores. All models' performance of the boolean query is very low because 94% of boolean queries have 0-2 positive facts, making it hard to conduct accurate retrieval on a large size of database.

Reader Performance We report the corresponding reader accuracy in Table 2. Because of the higher variation in reader performance, we report the standard deviations in the table. Compared to the SSG, VKGFR shows better answer accuracy, indicating that our more precise retriever leads to performance improvements on the reader. In comparison to the OPQL and DensePhrase, VKGFR shows better total accuracy. The answer candidates of the boolean query are easier than the other queries, so the accuracy is much higher for all models, even TOME, compared to other question types. The count query exhibits the lowest accuracy compared to the others due to its requirement of accurately predicting every positive sample.

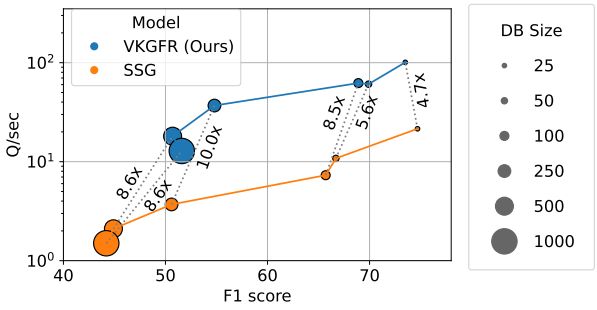


Figure 6: Inference speed and the retrieval performance of SSG (orange circles) and VKGFR (blue circles) on different sizes of the WikiNLDB test set. The x-axis represents the retrieval F1 score, and the y-axis represents the inference speed. The size of the circles indicates the database sizes. The numbers on the circles indicate the ratio of q/sec between two models on the same DB size.

5.3 Computational Efficiency

We measure the number of queries that each retriever can process in a second (Q/sec); the inference speed is measured by one Quadro RTX A6000 48GB GPU. We plot the speed-accuracy trade-off with our model and the SSG baseline in Figure 6. The speed of all models includes the time required for query embedding. VKGFR showed at least 8.9 times faster inference speed than the SSG on all database sizes and a higher F1 score on DB size larger than 25, which is more representative of real-world applications. Table 3 shows the retrieval speed of each model on the largest database size (1000 facts). VKGFR models can conduct efficient retrieval with the simple model structure compared to the other transformer-based models. To perform inference on WikiNLDB on TOME, MDPR, and SSG, a new query vector must be encoded for each reasoning step. For example, based on the SSG, 440 BERT encodings are required per query in DB

Model	Min/Max			Set			Count			Bool			Total		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
TOME	17.5	72.5	28.2	15.1	77.5	25.2	14.8	80.5	25.0	2.6	90.9	5.0	15.1	77.2	25.3
MDPR	25.8	72.4	37.7	23.2	77.9	35.3	22.5	79.9	34.9	15.8	77.9	26.2	23.5	76.2	35.7
SSG	35.0	88.3	50.1	27.7	84.7	41.7	26.8	84.6	40.6	19.2	82.4	31.1	29.8	85.9	44.2
VKGFR (Ours)	44.3	70.5	**54.4**	37.8	76.3	**50.5**	39.3	76.9	**52.0**	17.5	82.8	28.9	39.4	74.6	**51.6**
w/ OPQL embedding	44.0	68.5	53.6	35.2	75.6	48.0	36.5	75.6	49.2	17.7	84.0	29.2	37.9	73.3	50.0
w/ DensePhrase embedding	41.6	68.8	51.8	37.3	72.8	49.3	39.8	74.0	51.7	19.8	83.0	**32.0**	38.5	72.2	50.2

Table 1: Precision, Recall, F1 score and retrieval speed of each retriever on the database size 1000. VKGFR shows the highest F1 score on Min/Max, Set, and Count type queries.

Model	Min/Max (std)	Set (std)	Count (std)	Bool (std)	Tot (std)
TOME	36.68 (0.75)	54.91 (2.81)	15.69 (3.81)	**86.75** (3.71)	38.97 (1.94)
MDPR	35.67 (1.28)	41.03 (1.53)	10.83 (1.97)	56.84 (1.77)	31.01 (0.43)
SSG	43.04 (0.44)	58.03 (1.74)	15.37 (1.78)	78.21 (3.06)	41.64 (0.99)
VKGFR (Ours)	44.92 (1.03)	**60.01** (0.86)	**17.80** (0.89)	83.23 (2.18)	**43.91** (0.59)
w/ OPQL embedding	**45.61** (1.44)	57.08 (0.94)	16.86 (0.49)	82.16 (0.81)	43.10 (0.23)
w/ DensePhrase embedding	41.30 (3.59)	55.39 (0.70)	**17.80** (0.49)	79.70 (0.74)	41.04 (1.64)

Table 2: Fine-tuned reader accuracy on each retriever's result for different types of queries on the database size 1000. The standard deviation is included in this table because the std of the reader results is bigger than the retrieval results' std.

Model	Speed (Q/sec)	F1
TOME	0.19	25.29
MDPR	0.63	35.67
SSG	1.46	44.21
VKGFR (Ours)	12.83	**51.59**
w/ OPQL embedding	**13.01**	49.95
w/ DensePhrase embedding	12.45	50.22

Table 3: Represents the speed of each model on the largest database size (1000) and corresponding retrieval F1 score.

DB Size	# of DBs	Avg Size/DB	Avg Indexing Time /DB
25	621	1MB	0.3s
50	499	2MB	0.6s
100	250	4MB	1.2s
250	100	10MB	3.3s
500	50	17MB	6.1s
1000	25	26MB	12.0s

Table 4: Represents the size of pre-indexes for test dataset on each database size, and taking time for the embedding queries and facts.

size 1000 on average, but VKGFR only needs 1 BERT encoding per query.

We included the amortized time for indexing our embeddings to ensure a fair comparison. However, this indexing includes additional storage overheads. We report these storage costs in Table 4.

6 Experiments and Results on MetaQA

Experiment Setup To verify VKGFR on other tasks, we experiment with MetaQA (Zhang et al., 2018), which is a multi-hop retrieval over a pre-defined knowledge base built on the WikiMovies dataset. Unlike WikiNLDB, the number of hops is prefixed before the inference, so there is no end prediction for this case. The questions of MetaQA are generated from predefined templates, and corresponding answers exist on the knowledge base. We fine-tune the relation encoder with MetaQA dataset as Sun et al. (2021) and use the same training & inference configuration as 4.1. For inference, we apply a sparse filter that the retrieved knowledge base should include the topic entity of the query for increasing the accuracy (Dhingra et al., 2020; Sun et al., 2021).

Results Table 5 reports the Hit@1 results of different models on the MetaQA dataset. The VKGFR outperforms the previous approach by at least 1.7 points in every case, indicating our VKG representation contains the essential information for the question-answering task more than others. The 1-hop performance of OPQL is not mentioned in the paper, but the authors said the performance is lower than the DrKIT.

Model	1Hop	2Hop	3Hop
KVMem	-	7.0	19.5
DrQA	55.3	32.5	19.7
GRAFT-Net	82.5	36.2	40.2
PullNet	84.4	81.0	78.2
DrKIT	84.4	86.0	87.6
OPQL	-	88.5	87.1
VKGFR	**86.1**	**93.7**	**92.1**

Table 5: Hit@1 results on MetaQA dataset. Each result is from the original paper.

7 Related Works

Building a semi-structured representation from textual sources has been an important direction in handling reasoning queries (Asai et al., 2020; Sun et al., 2019; Dhingra et al., 2020). This is because reasoning tasks often require entity matching, and previous dense retrieval methods are insufficient for entity representation learning. For this reason, many studies have focused on entity-matching-based retrieval methods (Sun et al., 2018, 2019; Cao et al., 2019). These studies find supporting facts by iteratively matching entities that appeared in a given question and documents, similar to human information-seeking processes. Furthermore, contextualized entity embedding methods have been proposed. These methods are specifically designed for entity representation and capture more fine-grained semantic meanings of entities (Lee et al., 2021; de Jong et al., 2021).

Inspired by previous entity-matching-based approaches, some studies propose to use relations between entities as well as entity vectors (Dhingra et al., 2020; Sun et al., 2021; Seonwoo et al., 2022). These approaches use a relation encoder to encode the semantic meaning of the relation of entities, then construct a graph consisting of entity vectors and their relation vectors. Dhingra et al. (2020) proposes a virtual knowledge base, which consists of trainable entity vectors. Sun et al. (2021) further develops this approach to use the relation between entities and propose a virtual knowledge graph (VKG), which consists of entity and relation vectors. Seonwoo et al. (2022) adopts the VKG to the domain-specific document retrieval with insufficient training data. Unlike the previous approach, our methods target variable length of multi-hop database reasoning and show the best performance.

8 Conclusions

In this paper, we propose **VKGFR** enable multi-hop retrieval with faster speed and high performance over the WikiNLDB dataset. Our multi-hop retrieval mechanism does not require re-embedding of facts, resulting in fewer queries to an encoder model and allowing it to take advantage of pre-indexed fact representations. VKGFR retrieves upon that pre-indexed VKG representation which is highly contributing to the faster inference speed. On other tasks, VKGFR shows the best performance on the general knowledge-base multi-hop QA dataset, MetaQA. This research demonstrates the applicability of VKG text representation in the task of multi-hop database reasoning.

Limitations

VKGFR retrieves the knowledge base that consists of explicit entities and relations. If the knowledge base becomes more complex, with no explicit entities and relations in the sentence, new VKG encoding methods will be required for good performance.

References

Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1):29–81.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–

2905, Florence, Italy. Association for Computational Linguistics.

Yu Cao, Meng Fang, and Dacheng Tao. 2019. BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 357–362, Minneapolis, Minnesota. Association for Computational Linguistics.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William Cohen. 2021. Mention memory: incorporating textual knowledge into transformers through entity mention attention. *ArXiv preprint*, abs/2110.06176.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William W. Cohen. 2022. Mention memory: incorporating textual knowledge into transformers through entity mention attention. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick 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)*, pages 6769–6781, Online. Association for Computational Linguistics.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6634–6647, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

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.

Yeon Seonwoo, Seunghyun Yoon, Franck Dernoncourt, Trung Bui, and Alice Oh. 2022. Virtual knowledge graph construction for zero-shot domain-specific document retrieval. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1169–1178, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. 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 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Haitian Sun, Patrick Verga, Bhuwan Dhingra, Ruslan Salakhutdinov, and William W. Cohen. 2021. Reasoning over virtual knowledge bases with open predicate relations. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9966–9977. PMLR.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021a. Database reasoning over text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3091–3104, Online. Association for Computational Linguistics.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021b. From natural language processing to neural databases. In *Proceedings of the VLDB Endowment*, volume 14, pages 1033–1039. VLDB Endowment.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016a. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016b. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076. AAAI Press.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Hyperparameters

We use the AdamW optimizer (Loshchilov and Hutter) with a warmup ratio of 0.1 in all our experiments, and use four Quadro RTX A6000 48GB GPUs for model training. Table 6 represents our training hyper-parameters. The relation encoder and NeuralSPJ are based on the bert-large and T5 respectively. We trained our model based on the hugging face transformers(Wolf et al., 2020). For the hyperparameters for relation encoder and NeuralSPJ, we followed the original paper. For the VKGFR, we experiement with different learning rate (5e-4, 1e-4) and choosed the best performing one. For the layer number, we experimented with 1,2,4,8 and choosed the best-performing one.

B Model Parameter Size

Table 7 represents the number of parameters of baseline models.

C Dataset

The WikiNLDB Data is available on GitHub[2] and we used the pre-splited train, valid and test data. The MetaQA dataset can also be found on GitHub[3], and we followed a similar methodology as WikiNLDB when working with it.

[2]https://github.com/facebookresearch/NeuralDB
[3]https://github.com/yuyuz/MetaQA

Model	Hyperparmeter	Value
Relation Encoder	Learning rate	2e-5
	Number of epochs (fine-tuning)	2
	Number of epochs (pre-tuning)	3
	Batch size per device	24
	Relation vector size	1024
VKGFR	Learning rate	5e-4
	Hidden dimension of linear layer	4096 * 512
	Layer number of rnn	1
	Dropout rate	0.2
	Number of epochs	20
	Batch size for device	16396
NeuralSPJ (Reader)	Learning rate	1e-4
	Number of epochs	3
	Batch size for device	8

Table 6: Hyperparameters of pre-training and fine-tuning the relation encoder

Model	Number of parameters
TOME	53,057,920
MDPR	125,238,274
SSG	66,362,880
VKGFR (Ours)	6,297,088

Table 7: Represents the number of parameters of baseline models