Alexandria Engineering Journal 130 (2025) 198–207

Contents lists available at ScienceDirect

Alexandria Engineering Journal

journal homepage: www.elsevier.com/locate/aej

Original article

LLM-powered database migration: A framework for knowledge graph system evolution

Shangqing Zhao [a],*, Qifan Zhang [a], Man Lan [a,b,c]

[a] School of Computer Science and Technology, East China Normal University, Shanghai, China
[b] Lab of Artificial Intelligence for Education, East China Normal University, Shanghai, China
[c] Shanghai Institute for AI Education, East China Normal University, Shanghai, China

ARTICLE INFO

Keywords:
Knowledge graph
Question answering
Large Language Model
In-context learning
Database migration

ABSTRACT

Database migration, particularly the translation of query languages, remains a significant barrier to modernizing data infrastructure. This challenge is especially acute as organizations adopt advanced knowledge graph (KG) technologies to support demanding applications in domains like smart cities and eHealth. This paper introduces a novel, LLM-powered framework for automated query translation, demonstrated through KG migration from RDF/SPARQL to LPG/Cypher. Our method leverages in-context learning with strategic exemplar selection and iterative refinement, achieving up to 89.6% translation accuracy and a 97.3% executable rate without requiring large parallel corpora or manual rule creation. Experiments on both the KQA Pro and enterprise-scale DBLP-QuAD datasets validate the approach's effectiveness and scalability. With migration costs under $1.50 for thousands of queries, our framework offers an economically viable solution that reduces migration costs and accelerates the adoption of modern database technologies for next-generation applications.

1. Introduction

Knowledge graphs (KGs) have become foundational to modern data-driven applications [1–3], powering advanced analytics, question answering, and intelligent systems across domains such as smart cities, eHealth, and enterprise knowledge management [4,5]. As the demands on these systems grow, organizations are increasingly migrating from traditional RDF-based KGs to more flexible and performant labeled property graph (LPG) databases like Neo4j [6,7]. This shift promises improved scalability, developer experience, and support for complex analytics. However, it also introduces a critical engineering challenge: migrating not only the underlying data, but also the vast ecosystem of queries and applications built atop these knowledge graphs.

While data model transformation tools (e.g., *Neosemantics*) can automate the conversion of RDF data to LPG formats, the translation of query languages — specifically from SPARQL to Cypher — remains a major bottleneck. Existing solutions typically rely on handcrafted rules or require large parallel corpora of paired queries, both of which demand significant manual effort and domain expertise [8,9]. These limitations make large-scale, cost-effective migration projects risky and inaccessible for many organizations, slowing the adoption of next-generation KG technologies.

To address this gap, we propose a novel, LLM-powered framework for automated query translation. Our approach leverages the in-context learning capabilities of large language models (LLMs), starting from a small, manually curated set of SPARQL–Cypher exemplar pairs. By employing semantic retrieval to select the most relevant exemplars for each input and iteratively expanding the exemplar database with verified translations, our system enables LLMs to master complex translation patterns—without explicit ontology injection or extensive parallel data.

This framework offers several key advantages:

- **Minimal manual effort:** It eliminates the need for extensive rule engineering or large annotated corpora, dramatically reducing migration costs.
- **Implicit ontology learning:** The LLM infers schema and ontology mappings directly from exemplars, enabling robust translation even in the absence of explicit schema information.
- **Scalability and economic viability:** Our experiments on both the KQA Pro benchmark and the enterprise-scale DBLP-QuAD dataset (252M triples) demonstrate high translation accuracy (up to 89.6%), a 97.3% executable rate, and total migration costs under $1.50 for thousands of queries.

Our contributions have direct implications for advancing the state-of-the-art in electronics engineering applications. In smart cities, our

* Corresponding author.
E-mail addresses: sqzhao@stu.ecnu.edu.cn (S. Zhao), qifanwz@gmail.com (Q. Zhang), mlan@cs.ecnu.edu.cn (M. Lan).

https://doi.org/10.1016/j.aej.2025.08.014
Received 8 April 2025; Received in revised form 12 June 2025; Accepted 9 August 2025
Available online 16 September 2025

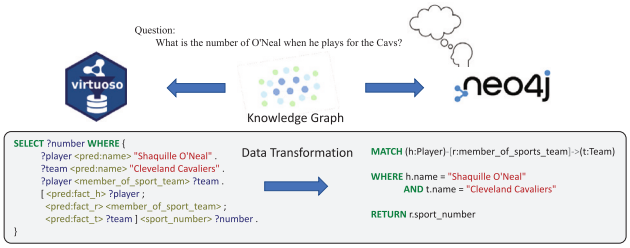

Fig. 1. An example of question-answering over knowledge graph. Virtuoso is commonly used for hosting SPARQL endpoints and Neo4j is a popular graph database. In this work, question and Cypher pairs are automatically transformed from existing SPARQL datasets via LLMs to achieve the migration of QA systems.

framework can facilitate the migration of legacy traffic management and sensor data systems to modern graph databases, enabling more sophisticated real-time analytics and cross-domain data integration. For eHealth applications, it enables seamless evolution of medical knowledge systems, supporting interoperability between different healthcare standards and facilitating advanced analytics for personalized medicine. In IoT and edge computing environments, where data schemas and system requirements evolve rapidly, our approach provides the agility needed to adapt knowledge graph infrastructures without extensive manual intervention.

The remainder of this paper is organized as follows: Section 2 reviews related work in database migration and knowledge graph question answering systems. Section 3 presents the technical preliminaries and formal problem definition. Section 4 details our LLM-powered iterative refinement framework. Section 5 provides comprehensive experimental evaluation on two datasets. Section 6 discusses practical considerations including cost analysis and limitations. Finally, Section 7 concludes with future research directions and broader implications for database system evolution.

2. Related work

2.1. Database migration and evolution

Database migration has become increasingly crucial as enterprises modernize their data infrastructure to meet evolving business needs. Traditional migration approaches typically focus on schema mapping and data transformation [10], often requiring substantial manual effort and domain expertise. While tools like AWS Database Migration Service and Oracle SQL Developer Migration Workbench have emerged to facilitate this process, they primarily address structured data migration between similar database types.

The emergence of NoSQL databases has introduced new challenges to database migration. Unlike traditional relational databases, NoSQL systems often have fundamentally different data models and query languages [11,12]. This heterogeneity makes automated migration particularly challenging, as it requires not only data transformation but also query and application adaptation. Recent research has explored various approaches to this challenge, including schema matching algorithms and query translation frameworks [8,9]. However, these methods often rely on rigid rules or patterns, limiting their adaptability to complex scenarios.

2.2. KGQA

KGQA aims to obtain knowledge from the KG to answer the given natural language question. In the early stages of research, the answer to a question can be an entity or relation in the KG. However, with the emergence of datasets focusing on complex questions, the answer can be a numerical result obtained through intricate aggregation operations. To solve such complex questions, recent KGQA approaches tend to generate logical forms through semantic parsing rather than relying on retrieval-ranking methods. Early approaches focused on building and converting Query Graphs into logical forms. With the advent of Pre-trained Models (PTMs) like BART [13], direct generation of logical forms from natural questions became feasible [14]. Most recently, Large Language Models (LLMs) have enabled new approaches, such as discriminative selection of candidate logical forms [15] and few-shot in-context learning [16,17], though some solutions still face challenges with computational costs.

A typical KGQA system includes a KG deployed in a Database Management System (DBMS) and a model that takes questions as input and outputs logical forms such as SPARQL [18], S-expressions [19], and CQL [20]. For example, if the model takes SPARQL as the target logical form, the DBMS should be database software which provides SPARQL query endpoints like Jena,[1] Virtuoso,[2] etc. The evolution of KGQA systems reflects a broader trend in database technology advancement, where systems need to adapt to new database paradigms while maintaining their core functionality.

2.3. LLMs in database applications

The emergence of Large Language Models (LLMs) has opened new possibilities in database management and evolution. Recent research has demonstrated LLMs' potential in various database-related tasks, including natural language to SQL translation, schema design, and data integration [21,22]. The key advantage of LLMs lies in their ability to understand both natural language and structured queries, making them particularly suitable for bridging different database paradigms.

In-context learning, a distinctive capability of LLMs, has shown promising results in various database applications. Unlike traditional machine learning approaches that require extensive training data, in-context learning enables models to adapt to new tasks with just a few examples [23]. This property is particularly valuable in database migration scenarios, where parallel training data is often scarce or expensive to obtain.

3. Preliminaries

3.1. Knowledge base

A typical knowledge base consists of an ontology $\mathcal{O}$ and a model $\mathcal{M}$ [24] . The ontology $\mathcal{O}$ defines the concepts, relationships, and constraints within a specific domain of knowledge and $\mathcal{M}$ is data model representing facts. In this work, the RDF graph model and Neo4j's labeled property graph (LPG) model are involved.

For RDF, $\mathcal{M} \subseteq (\mathcal{E} \cup \mathcal{R}) \times \mathcal{R} \times (\mathcal{C} \cup \mathcal{E} \cup \mathcal{V})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of binary relations, $\mathcal{C}$ is a set of classes, and $\mathcal{V}$ is a set of literal values. It is noteworthy KQA Pro contains qualifier knowledge so that a relationship can also be the head of the triple as shown in Fig. 2.

For LPG, $\mathcal{M} \subseteq \mathcal{N} \times \mathcal{R} \times \mathcal{P} \times \mathcal{L}$, where $\mathcal{N}$ is a set of nodes, $\mathcal{R}$ is a set of directed relations, $\mathcal{P}$ is a set of properties, and $\mathcal{L}$ is a set of labels. Both nodes and relationships possess distinctive identifiers and can store properties represented as key–value pairs. Nodes can be labeled to be grouped. The edges in LPG representing the relationships always have a start node and an end node, making the graph a directed graph.

Fig. 2 gives an example of Neo4j's labeled property graph model and RDF data model. The two models store the same knowledge, but require different logical forms to answer the question *"What is the number of O'Neal when he plays for the Cavs?"* as shown in Fig. 1 . The evolution

[1] https://jena.apache.org/.
[2] https://github.com/openlink/virtuoso-opensource.

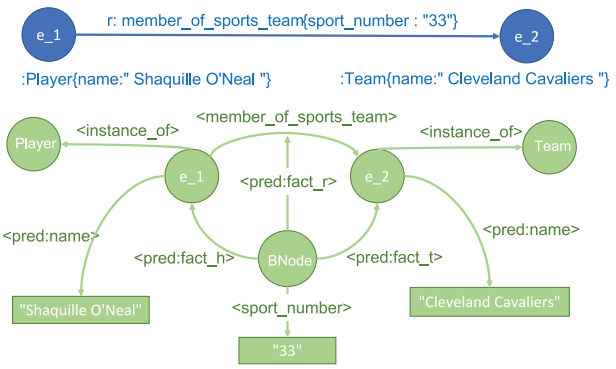

Fig. 2. An example of the LPG model (blue part) and RDF graph model (green part). *sport_number* is a relation qualifier of relation *member_of_sports_team*. RDF graph requires a blank node (BNode) to describe the qualifier knowledge (the fact that when O'Neal plays as a member of Cleveland Cavaliers, his sport number is 33).

from RDF to LPG reflects a broader trend in database technology advancement, where systems adapt to meet changing requirements for data expressiveness, query performance, and maintenance efficiency. This transition, however, necessitates comprehensive migration strategies that address both data transformation and query adaptation.

3.2. Query languages

SPARQL (SPARQL Protocol and RDF Query Language) and Cypher represent two distinct approaches to graph querying, each optimized for its respective data model. SPARQL, designed for RDF graphs, uses a pattern-matching approach based on triple structures. A typical SPARQL query begins with keywords like SELECT or ASK, followed by graph pattern expressions that can include variables (e.g., `?player`, `?team`). The language supports complex operations through keywords such as FILTER, UNION, and EXISTS. Cypher, Neo4j's query language, adopts a more visual approach to graph pattern matching. Patterns in Cypher use ASCII-art style syntax, where nodes are represented by parentheses and relationships by arrows (e.g., `(h:Player)-[:member_of_sport_team]->(t:Team)`). This visual representation often makes Cypher queries more intuitive for developers, particularly when dealing with complex graph patterns. The language includes features like property access, pattern matching, and aggregation functions, making it well-suited for modern graph database applications.

The fundamental differences between these query languages reflect their underlying data models and design philosophies. While SPARQL excels at handling semantic web data and standardized knowledge representations, Cypher provides more intuitive and efficient querying for property graph applications. This divergence creates unique challenges for system migration, requiring sophisticated translation mechanisms that preserve both semantic meaning and query efficiency.

3.3. Task definition

Given a source database system S with its query language L_s and a target database system T with its query language L_t, the database migration task involves: (1) Data Migration: Converting data from S to T while preserving data integrity and relationships; (2) Query Transformation: Transforming queries written in L_s to equivalent queries in L_t; (3) Application Adaptation: Modifying applications to work with the new query language and database system.

In this paper, we demonstrate our approach through a knowledge graph migration scenario, where the source system S is RDF, the target system T is Neo4j, and convert $D_s = \{(q_i, s_i) \mid i \in N\}$ to obtain $D_c = \{(q_i, c_i) \mid i \in N\}$. For query transformation, $D_s = \{(q_i, s_i) \mid i \in N\}$ is the original QA pairs with natural language questions q_i and SPARQL queries s_i, the goal is to generate equivalent Cypher queries c_i to obtain $D_c = \{(q_i, c_i) \mid i \in N\}$. While tools like *neosemantics* handle the data migration aspect (converting RDF to LPG), the key challenge lies in query transformation. Traditional approaches often require extensive manual effort or large parallel corpora for training. We propose leveraging LLMs through in-context learning to automate this process, potentially opening new possibilities for broader database migration scenarios.

4. In-context data transformation

Our approach leverages in-context learning capabilities of LLMs to facilitate database query translation. The key idea is to provide LLMs with carefully selected exemplars of query pairs (SPARQL–Cypher) that demonstrate the translation patterns. As illustrated in Fig. 3, our method consists of three main components: (1) Exemplar Creation: Manual annotation of a small seed set of SPARQL–Cypher query pairs. (2) Exemplar Selection: Strategic selection of relevant exemplars for each input query. (3) Iterative Refinement: Continuous expansion of the exemplar database through verified transformations.

4.1. Prompt design

The effectiveness of in-context learning heavily depends on prompt design. The ontology information seems to be very important in the task. However, the ontology information may be too much text, confusing the LLM and also increasing the API cost. We give a simple instruction and omit the explicit ontology information. Our prompt template is shown in Fig. 4. We add special tokens $< soe >$ and $< eoe >$ to represent the start and the end of an exemplar. For the target question, the LLM will complete the Cypher query which is leaving empty in the prompt.

4.2. Exemplar selection strategies

As we omit the ontology information in the instruction, LLM has to capture it from the in-context exemplars. To verify this idea, we explore two strategies for selecting in-context exemplars:

Random Selection: This baseline strategy randomly samples k exemplars from the available SPARQL–Cypher pairs. While simple, this approach helps establish the lower bound of performance and demonstrates the robustness of LLM-based translation.

Semantic Retrieval: We employ BM25 algorithm [25] to retrieve exemplars that are semantically similar to the input query. The similarity is computed based on:

$$score(Q, E) = BM25(Q.question, E.question)$$

where Q is the input query and E is a candidate exemplar. This approach ensures that the selected exemplars demonstrate translation patterns relevant to the current query. This semantic retrieval can be replaced or combined with neural retrieval methods, such as BERT [26] and SentenceTransformers [27].

4.3. Iterative refinement process

To improve the system's performance over time, we implement an iterative refinement process as shown in Algorithm 1. The iterative refinement process consists of four key phases:

Phase 1: Initial Translation. The system generates Cypher queries for input SPARQL queries using the current exemplar database. Each translation attempt leverages the in-context learning capabilities of the LLM with retrieved relevant exemplars.

Phase 2: Verification. Generated Cypher queries undergo rigorous verification through:

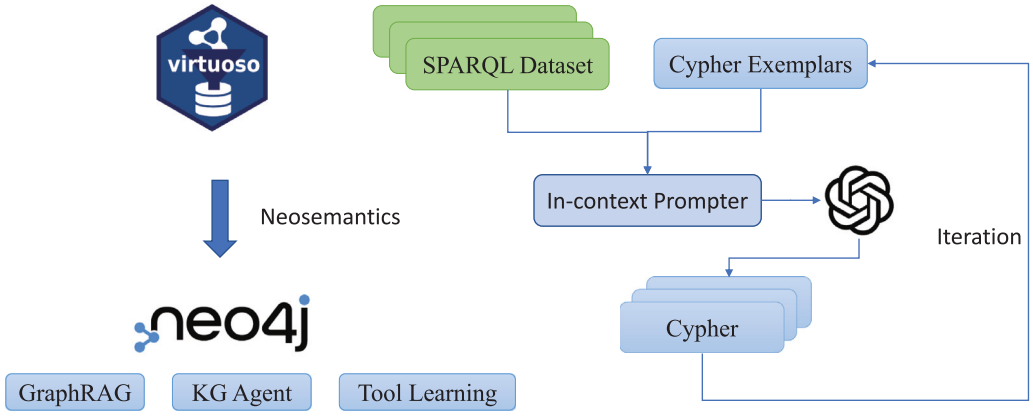

Fig. 3. Overview of our proposed migration pipeline. The KG is transferred from Virtuoso to Neo4j and application dataset is transformed from SPARQL to Cypher.

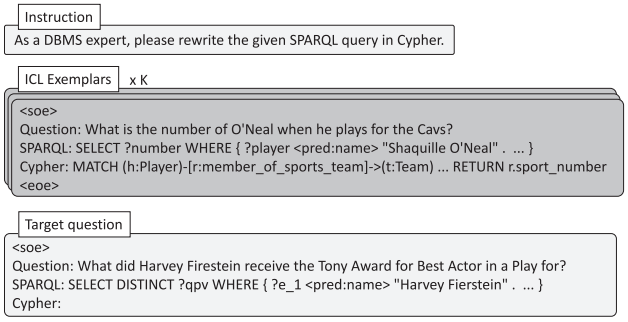

Fig. 4. The prompt design for in-context generation.

Algorithm 1 Iterative Refinement Process

Require: D_{seed} ▷ Initial seed set of SPARQL–Cypher pairs
Require: Q_{new} ▷ New SPARQL queries to translate
Require: k ▷ Number of exemplars to use
Ensure: D_{expanded} ▷ Expanded exemplar database

function IterativeRefinement($D_{\text{seed}}, Q_{\text{new}}, k, \theta$)
  $D_{\text{expanded}} \leftarrow D_{\text{seed}}$
  for each query q in Q_{new} **do**
    $E_q \leftarrow \text{SelectExemplars}(D_{\text{expanded}}, q, k)$
    $c_q \leftarrow \text{GenerateTranslation}(q, E_q)$ ▷ Using LLM
    $result \leftarrow \text{VerifyTranslation}(c_q)$
    if $result$ is True **then**
      $D_{\text{expanded}} \leftarrow D_{\text{expanded}} \cup \{(q, c_q)\}$
    end if
  end for
  return D_{expanded}
end function
function VerifyTranslation(c_q)
  Execute c_q against target database
  Compare results with source query execution
  return result
end function

- Syntax validation against Neo4j's query parser
- Execution against the target database
- Result comparison with original SPARQL query outputs

Phase 3: Exemplar Database Expansion. Successfully verified translations are incorporated into the exemplar database. A translation is considered successful when it:

- Maintains syntactic correctness
- Produces equivalent results to the source query
- Preserves the semantic intent of the original query

Phase 4: Iterative Enhancement. The expanded exemplar database feeds back into the translation process, creating a continuous improvement cycle. We expect this iterative approach to enhance retrieval quality and form better ICL prompt quality, thus improving the translation accuracy from accumulated successful patterns.

The effectiveness of this process is demonstrated through empirical evaluation in Section 5, where we observe consistent improvements in both translation accuracy and query execution success rates across iterations.

5. System migration experiment

Our experiments address three key research questions: (1) How effectively can LLMs translate between database query languages without extensive training? (2) What impact do different exemplar selection strategies have on translation quality? (3) How does the iterative refinement process improve performance over time? To answer these questions, we designed a comprehensive evaluation framework using knowledge graph query translation as a representative database migration task.

5.1. Experimental settings

5.1.1. Datasets

We evaluate our approach on two datasets of varying scale and complexity to demonstrate its robustness:

KQA Pro: We use KQA Pro [14] as our primary evaluation benchmark—an open-domain KGQA dataset featuring complex questions requiring multi-step reasoning. The dataset provides annotated programs for each question, representing the reasoning steps required to derive the answer. We use the number of steps in these programs, which we refer to as "program length", as a proxy for question complexity in our analysis. The underlying knowledge graph contains approximately 20,000 entities, 1200 relations, and 900,000 triples merged from Freebase and Wikidata.

DBLP-QuAD: To validate scalability, we conducted experiments on DBLP-QuAD [28], built on the significantly larger DBLP scholarly knowledge graph (2.9 million person entities, 6 million publication entities, and over 252 million RDF triples).

```
CREATE CONSTRAINT n10s_unique_uri ON (r:Resource)
ASSERT r.uri IS UNIQUE;

CALL n10s.graphconfig.init({
  handleVocabUris: "SHORTEN",
  handleMultival: "ARRAY",
  handleRDFTypes: "LABELS_AND_NODES"
})

CALL n10s.rdf.import.fetch(
  "file:///root/kqa_kb.ttl",
  "Turtle"
)
```

Fig. 5. Knowledge graph migration configuration: This Cypher script shows the initialization process for transferring RDF data to Neo4j using the neosemantics plugin, illustrating the data transformation component of our migration framework.

Table 1
Dataset statistics: The answers in KQA Pro are not available, so we take 1000 samples from the validation set for evaluation. The answers in DBLP-QuAD are available, so we use the original test set.

	Train	Validation	Test
KQA Pro	94,376	11,797	11,797
KQA Mini	50	–	1000
DBLP-QuAD	7000	1000	2000
DBLP Mini	90	–	2000

For data migration, we used *neosemantics* to convert RDF data to Neo4j's LPG format, with the configuration shown in Fig. 5. This configuration defines the ontology mapping rules from RDF to LPG. For query translation evaluation, we created KQA Mini—a dataset containing 50 samples from the training set and 1000 from the validation set (SPARQL–Cypher pairs are available), maintaining the distribution of 12 question types from the original dataset (Table 1). For DBLP-QuAD, we created a smaller dataset DBLP Mini containing 90 samples from the training set with manually annotated SPARQL–Cypher pairs and use the original test set (no Cypher annotations are available) for evaluation.

5.1.2. Evaluation metrics

We employ four complementary metrics to assess query translation quality from multiple perspectives:

Answer Accuracy/F1: The primary metric measuring whether the translated query produces the same answer as the original query when executed against the respective databases. For KQA Pro, where answers are typically single values, we use accuracy:

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(a_i = \hat{a}_i) \tag{1}$$

where a_i is the expected answer and $\hat{a}_i$ is the answer produced by the translated query. For DBLP-QuAD, where answers can contain multiple values, we use F1 score to measure the overlap between predicted and golden answer sets following the DBLP-QuAD paper [28]:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2}$$

where $Precision = \frac{|A_{pred} \cap A_{gold}|}{|A_{pred}|}$ and $Recall = \frac{|A_{pred} \cap A_{gold}|}{|A_{gold}|}$. Here, A_{pred} and A_{gold} are the sets of predicted and gold standard answers respectively.

Translation Fidelity (BLEU-4): Measures syntactic similarity between the generated and reference translations [29]:

$$BLEU = min(1, 1 - \frac{r}{c}) \cdot \exp(\sum_{n=1}^{N} w_n \log p_n) \tag{3}$$

where r is reference length, c is candidate length, and p_n is n-gram precision.

Tree Edit Distance (Tree-ED): Measures structural similarity between the generated and reference Cypher queries by comparing their Abstract Syntax Trees (ASTs). We first parse the Cypher queries into ASTs and then calculate the tree edit distance [30,31]. The similarity score is normalized using:

$$Tree - ED = \exp(-\frac{edit\_distance}{max\_distance}) \tag{4}$$

where $edit\_distance$ is the minimum number of operations required to transform one AST into another, and $max\_distance$ is the maximum possible edit distance between the trees. If generated query is syntactically incorrect (cannot be parsed into an AST), the Tree-ED score is 0. If $edit\_distance$ equals to 0, the Tree-ED score is 1. This metric helps assess how well the generated query matches the reference query in terms of the query structure.

Executable Rate (ER): Measures syntactic correctness—the percentage of translated queries that can be parsed and executed by the target database:

$$ER = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{y}_i) \tag{5}$$

where $f(\boldsymbol{y}_i)$ returns 1 if query $\boldsymbol{y}_i$ is executable and 0 otherwise.

These metrics provide a comprehensive assessment of both functional correctness (Acc/F1), structural similarity (Tree-ED), syntactic correctness (ER), and translation quality (BLEU-4).

5.1.3. Implementation details

All experiments used OpenAI's *GPT-4o-mini* model with temperature set to 0 for deterministic outputs. For the retrieval component, we implemented both BM25 and vector-based retrieval using SentenceTransformers.[3] The initial exemplar database contained 50/90 manually created SPARQL–Cypher translation pairs derived from the KQA Pro/DBLP-QuAD training set.

For the BART baseline, we fine-tuned a BART-large model [13] on our translation task, treating it as a sequence-to-sequence problem where SPARQL queries serve as input sequences and Cypher queries as target sequences. The model was fine-tuned using the train set used for the LLM exemplar database to ensure fair comparison. We used standard hyperparameters for sequence-to-sequence fine-tuning with a learning rate of 3e−5, batch size of 8, and trained for 50 epochs. This baseline represents a traditional neural machine translation approach to query translation, requiring explicit fine-tuning on parallel data.

5.2. Results on KQA

Table 2 presents our main experimental results on KQA:

Limitations of Traditional Seq2Seq Models The BART seq2seq model [13], despite achieving reasonable BLEU scores (71.22%), shows severe limitations in executable rate (38.30%) and answer accuracy (5.70%). The Tree-ED score of 70.72% indicates that while the model can generate structurally similar queries, it struggles to maintain the correct query grammar. This stark performance gap highlights the fundamental challenge traditional seq2seq models face with complex query translations, likely due to their inability to effectively capture the semantic intricacies and syntactically requirements of different query languages without substantial parallel training data.

Impact of In-Context Learning The substantial performance gap between zero-shot (13.20% Acc, 75.00% Tree-ED) and in-context learning with randomly selected examples (47.00% Acc, 85.04% Tree-ED) demonstrates that even basic exemplars significantly improve both

[3] The model used is *all-MiniLM-L6-v2*.

Table 2
Performance comparison across methods on KQA: The table shows progressive improvement in all metrics from zero-shot to iterative refinement approaches, with semantic retrieval (BM25) consistently outperforming random selection.

Method	BLEU	Tree-ED	ER (%)	Ans Acc (%)
bart [13]	71.22	70.72	38.30	5.70
zeroshot	39.25	75.00	91.00	13.20
random_k5	79.03	85.04	83.80	47.00
vector_k5	83.90	88.75	88.00	63.20
bm25_k5	86.79	89.91	88.80	66.80
bm25_k5_iter1	88.24	89.91	94.30	82.90
bm25_k5_iter2	88.42	89.91	96.10	86.70
bm25_k5_iter3	88.68	89.91	97.30	88.80
bm25_k5_iter4	88.58	89.91	96.90	89.60

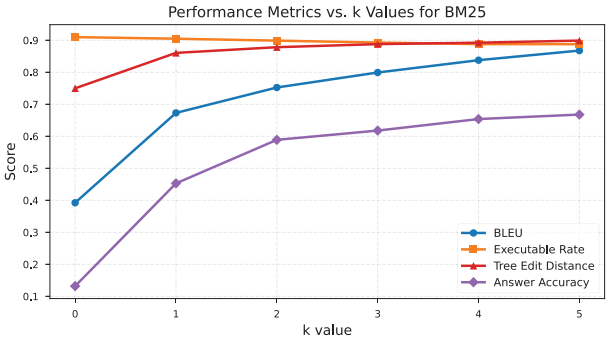

Fig. 6. The results on KQA when provided with different in-context exemplars.

translation quality and structural preservation. This confirms our hypothesis that LLMs can effectively learn database-specific patterns from a small set of examples, making them particularly suitable for database migration scenarios where parallel data is limited.

Retrieval Strategy Effectiveness Semantic retrieval using BM25 (66.80% Acc, 89.91% Tree-ED) outperforms random selection (47.00% Acc, 85.04% Tree-ED) by a substantial margin (+19.80% Acc, +4.87% Tree-ED), highlighting the importance of presenting relevant examples to the model. This improvement suggests that even simple lexical similarity-based retrieval methods can significantly enhance both translation quality and structural preservation in in-context learning for database query translation.

Number of ICL Exemplars The quantity of exemplars provided in the context is an intuitive factor that affects the final performance for in-context learning. We retrieve the K most similar samples to the question using BM25 [25] to form context, where K ranges from 1 to 5. The experimental results are illustrated in Fig. 6. The BLEU score shows consistent improvement as K increases, from 39.25% (zero-shot) to 86.79% (K = 5), indicating that additional examples help LLMs better understand the query translation patterns. The most significant jump occurs between zero-shot and K = 1 (from 39.25% to 67.32%), suggesting that even a single well-chosen example can substantially improve translation quality. The executable rate remains consistently high (around 89%) across different K values, with slight fluctuations. The Tree-ED starts from 75% and plateaus at around 89%. This suggests that LLMs maintain strong syntactic understanding regardless of the number of examples, likely due to their pre-training on programming languages. The answer accuracy shows significant improvement from zero-shot (13.2%) to K = 4 (65.4%), but plateaus afterwards. This pattern suggests that while more examples generally help, there might be a point of diminishing returns, possibly due to context window limitations or increased complexity in processing multiple examples.

These findings have important implications for database migration applications: while more examples generally improve performance, practitioners should balance the benefits of additional examples against computational costs and context window limitations. The high ER and Tree-ED across all settings also suggests that LLMs could be reliable tools for maintaining syntactic correctness in database query translation tasks.

Table 3
DBLP-QuAD results: Performance metrics on the larger DBLP scholarly knowledge graph (252M triples) showing consistent improvements across iterations, validating the approach's scalability to enterprise-scale knowledge bases.

Method	F1 score (%)	Executable (%)	Improvement (%)
bart [13]	21.41	90.00	–
zeroshot	11.65	92.00	–
random_k5	35.66	84.40	+24.01
vector_k5	37.16	86.40	+25.51
bm25_k5	45.60	89.55	+33.95
bm25_k5_iter1	66.07	92.15	+54.42
bm25_k5_iter2	69.59	92.80	+57.94
bm25_k5_iter3	70.56	92.30	+58.91
bm25_k5_iter4	70.91	92.05	+59.26

Iterative Refinement Benefits The progressive improvement across iterations (66.80% → 82.90% → 86.70% → 88.80% → 89.60% for Acc, while maintaining Tree-ED at 89.91%) validates our iterative refinement approach. Each iteration contributes meaningfully to performance, with the system becoming increasingly effective as the exemplar database expands with verified translations. Notably, the executable rate reaches 97.30% after three iterations, indicating high syntactic reliability suitable for production environments. The stable Tree-ED score across iterations suggests that our approach consistently preserves query structure while improving translation accuracy.

5.3. Results on DBLP-QuAD

Results on the DBLP-QuAD dataset (Table 3) demonstrate the approach's effectiveness on larger knowledge graphs. Despite DBLP containing approximately 100 times more entities than KQA Pro, our system achieves consistent improvements across iterations, with F1 scores increasing from 11.65% (zero-shot) to 70.91% (after four iterations). The executable rate remains high (>90%) throughout, indicating robust syntactic understanding regardless of knowledge graph scale.

Interestingly, BART achieves a relatively high executable rate (90.00%) but modest F1 score (21.41%) on DBLP-QuAD, suggesting that while it can produce syntactically valid queries for this dataset, it struggles with semantic correctness. This contrasts with its performance on KQA Pro, where it fails on both executable rate and answer accuracy, highlighting how dataset characteristics can significantly impact traditional seq2seq performance.

It is worth noting that the F1 scores on DBLP-QuAD are notably lower than the accuracy scores achieved on KQA Pro. This performance gap can be attributed to several fundamental challenges. First, DBLP-QuAD presents significant scale challenges with its 252M triples and approximately 9M entities, including numerous namesake scholars who share identical names but represent different academic entities, making precise entity disambiguation considerably more difficult. Second, the scholarly domain requires more complex schema and query structures, demanding specialized domain knowledge about academic relationships between authors, publications, venues, and citations that may be underrepresented in the GPT-4o-mini model's pre-training data. Third, according to the DBLP-QuAD paper [28], the test set is not independently and identically distributed with the training set, containing question types and patterns entirely absent from the training data. This non-IID setting creates additional difficulties for our *GPT-4o-mini* model when handling novel query structures and relationships. Using more powerful language models with stronger generalization capabilities

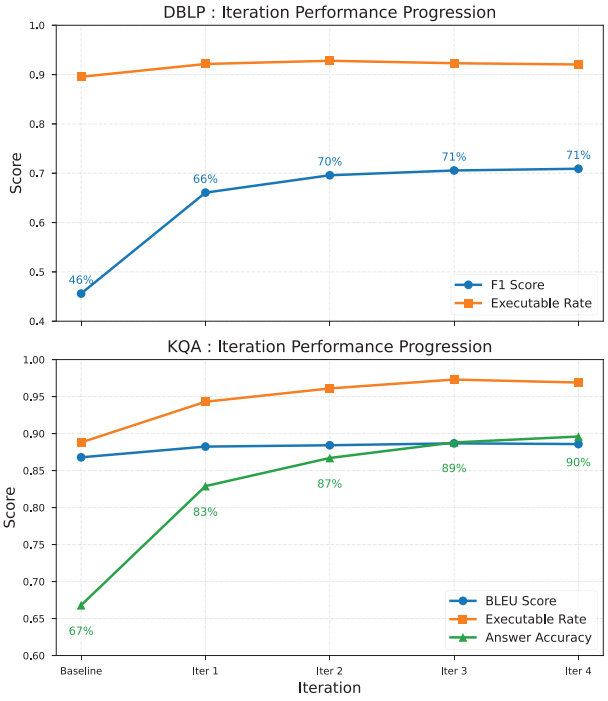

Fig. 7. Progressive performance improvement: This figure shows the consistent growth in both datasets across iterations, demonstrating the system's ability to continuously improve through feedback and exemplar database expansion.

might yield improved results on these challenging out-of-distribution questions. Despite these obstacles, the consistent improvement pattern across iterations demonstrates our approach's effectiveness even for complex, large-scale knowledge graphs, validating its potential for enterprise database migration scenarios.

The progressive improvement visualized in Fig. 7 demonstrates the system's continuous learning capability through the iterative feedback cycle, with each iteration contributing meaningfully to overall performance. It is noted that the rate of change in answer accuracy/F1 score does not align with that of Executable rate and BLEU score. When in-context learning is applied, both the Executable rate and BLEU score already achieve high performance, consistent with the Tree-ED results in Table 2, which remain at around 89.91%. During the iterative process, the LLM is able to acquire more ontology knowledge by retrieving new content from an updated exemplar pool, enabling it to make fine-grained adjustments to the generated queries so that they can yield correct answers upon execution.

6. Discussion

Our experiments demonstrate that an LLM-powered, iterative framework can effectively automate the complex task of query translation for database migration. The following sections explore the underlying reasons for its success, analyze a representative failure case, and discuss practical considerations such as cost and security. This analysis provides deeper insights into the method's mechanics, its current limitations, and its potential for real-world deployment in demanding application contexts like smart cities and eHealth.

6.1. Why and how our method works

To find out why and how our proposed method works, we grouped the samples in the KQA dataset into three groups based on the complexity of the question. The annotated program uses several functional step

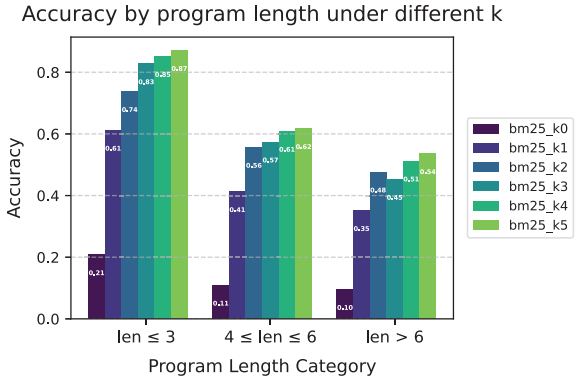

Fig. 8. Accuracy improvements across number of in-context exemplars (k value). *bm25_k0* refers to zeroshot setting in Table 2.

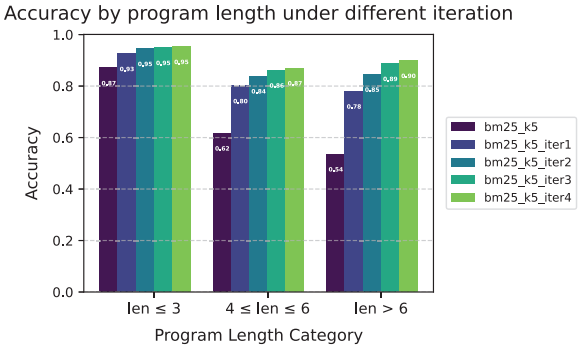

Fig. 9. Accuracy improvements across iterations.

to solve the question, the more steps, the more complex and difficult the question is. We group the samples into three groups based on the program length. Fig. 8 shows the accuracy improvements across different number of in-context exemplars (k value). Fig. 9 shows the accuracy improvements across iterations.

As shown in Fig. 8, the more complex a question is, the lower its accuracy tends to be. Across all three groups, the trend of accuracy (acc) as k varies is similar to that observed in Fig. 6. As k increases, performance improves across questions of varying difficulty levels—not just for simple questions. This indicates that the in-context learning approach demonstrates strong generalization across problems of different difficulty levels.

From Fig. 9, it can be seen that as the number of iterations increases, performance improves for questions of all difficulty levels, with a more pronounced improvement observed for difficult questions. This suggests that under the 5-shot in-context learning setting using the BM25 algorithm, most simple questions already achieve relatively high initial performance. Our iterative refinement process, however, is particularly effective in enhancing the model's performance on complex and long-form questions.

6.2. Exemplar selection methods

The selection of in-context exemplars plays a crucial role in few-shot learning performance. Our experiments demonstrate that different sets of exemplars can lead to significant variations in model performance, with observed accuracy differentials reaching 47.7% between random sampling and fourth iteration results. This observation aligns with recent findings in research [32,33], suggesting that the quality and relevance of in-context exemplars are as important as the model capabilities themselves.

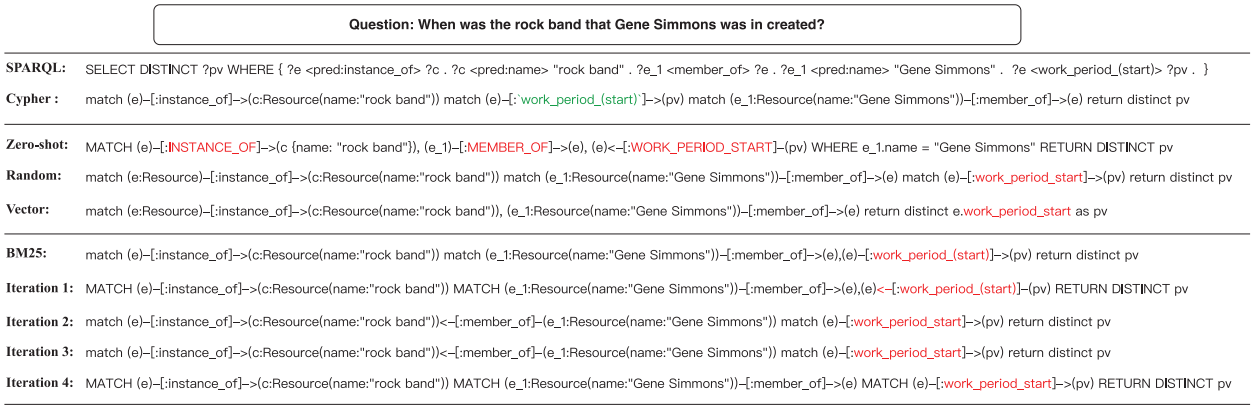

Fig. 10. A representative failure case. With in-context exemplars, the LLM performs well but fails to enclose the relation identifier in backticks (e.g., `` `relation` ``), a syntactic requirement it has not seen in the exemplars.

Currently, our approach employs a simple similarity-based retrieval method using traditional BM25. While this method shows promising results, there remains substantial room for improvement. Future work could explore more sophisticated example selection strategies, such as:

- **Hybrid Retrieval**: Combining semantic similarity with structural matching of question patterns and knowledge graph topology.
- **Dynamic Selection**: Adaptively choosing examples based on the complexity and characteristics of the input question.
- **Diversity-aware Sampling**: Ensuring selected examples cover different reasoning patterns while maintaining relevance.

These observations point to an important direction for future research: developing more principled approaches to example selection that consider both semantic relevance and structural characteristics of KGQA tasks. Such improvements could lead to more robust and efficient few-shot learning systems for complex reasoning tasks.

6.3. Case study

To better illustrate the working mechanism and limitations of our proposed method, we present a detailed analysis of a representative failure case, as shown in Fig. 10. We compare the outputs of all baseline methods for this specific example to reveal the strengths and weaknesses of each approach.

When directly prompting the LLM to generate a Cypher query for the migrated graph database (zero-shot setting), the model lacks knowledge of the ontology mapping from RDF to Neo4j. As a result, it incorrectly generates all relation identifiers in uppercase, which does not conform to the actual schema of the target database. This highlights the necessity of providing ontology-related information, either explicitly or implicitly, for accurate query translation.

With the random exemplar selection strategy, the LLM is able to produce a structurally correct Cypher query. This demonstrates that even randomly chosen in-context examples can help the model capture essential translation patterns, though the quality and relevance of the exemplars are not guaranteed.

In the dense vector retrieval setting, the LLM makes a different type of error: it incorrectly treats `work_period_start` as a node property rather than a relation identifier. This suggests that while dense retrieval can surface semantically similar examples, it may not always retrieve structurally appropriate ones, leading to subtle but critical mistakes in the generated queries.

The BM25-based retrieval setting yields results that largely preserve the correct structure of the Cypher query. However, both the BM25 and the first iteration of the iterative refinement process exhibit a common issue: the relation identifiers contain parentheses, which is not the correct Cypher syntax. Moreover, in the first iteration, the direction of the relation is generated incorrectly, further affecting the query's correctness.

Notably, the outputs of Iteration 2 and Iteration 3 are identical. This is because the exemplar pool did not change significantly between these iterations, resulting in BM25 retrieving the same top-5 examples for in-context learning. In Iteration 4, the model makes further fine-grained adjustments, recognizing that parentheses are not appropriate for relation identifiers in Cypher. However, it still fails to enclose the relation identifier in backticks (`), which is required for certain special identifiers in Cypher. This oversight is likely due to the absence of such cases in the initial seed pairs; the LLM has not seen an exemplar demonstrating the use of backticks for relation identifiers.

This case study underscores the importance of high-quality, diverse exemplars in the seed pool. Including a seed pair that demonstrates the use of backticks for special relation identifiers would likely enable the LLM to generalize this pattern to similar cases. Alternatively, employing a more powerful language model with stronger generalization capabilities could also address this issue. Overall, this analysis highlights both the promise and the current limitations of LLM-based query translation, and points to concrete directions for future improvement, such as targeted exemplar augmentation and model enhancement.

Table 4
The costs of the experiments. The price of gpt-4o-mini is $0.15 per million input token and $0.6 per million output token.

Experiment	Samples	Avg input	Avg output	Total input	Total output
KQA total cost: $0.41					
BM25 k = 5	1000	1195.8	88.9	1,195,759	88,889
Iteration 1	332	1166.5	103.0	387,277	34,209
Iteration 2	171	1181.7	99.7	202,069	17,050
Iteration 3	133	1202.2	108.5	159,895	14,425
Iteration 4	112	1203.4	102.9	134,783	11,526
Total	1748	1189.8	95.0	2,079,783	166,099
DBLP total cost: $1.04					
BM25 k = 5	2000	1016.8	79.1	2,033,644	158,254
Iteration 1	1112	1063.7	79.2	1,182,794	88,078
Iteration 2	703	1072.6	85.4	754,015	60,064
Iteration 3	632	1071.0	85.1	676,877	53,755
Iteration 4	614	1073.7	88.9	659,227	54,570
Total	5061	1048.5	82.0	5,306,557	414,721

6.4. Migration cost

The cost analysis (see Table 4) reveals several important insights about the economic feasibility of our approach:

Total Costs: The total cost for the KQA Pro experiments was $0.41, while the DBLP-QuAD experiments cost $1.04. These costs are reasonable considering the complexity of the task and the quality of results achieved.

Cost Distribution: For both datasets, the initial BM25 k = 5 phase accounts for the largest portion of the total cost, as it processes all samples. Subsequent iterations process fewer samples that failed to yield correct answers in the previous iteration.

Token Usage: The average input tokens per sample remain relatively stable across iterations (around 1200 for KQA Pro and 1050 for DBLP-QuAD), while output tokens show some variation but generally stay within a reasonable range (80–110 tokens per sample). The context length is far less than the limit of current powerful LLMs.

These cost statistics demonstrate that our approach is economically viable for real-world database migration scenarios, given that the cost of LLMs is decreasing rapidly.

6.5. Privacy and security considerations

While LLMs show promising capabilities in database migration tasks, their deployment in production environments raises important privacy and security concerns. The use of external LLM services requires sending database queries and schema information to third-party providers, potentially exposing sensitive business logic and data patterns. Our current approach partially mitigates these risks by only sharing query structures rather than actual data. However, even query patterns could reveal sensitive information about database design and business operations. Future implementations should consider several security measures: First, on-premise deployment of smaller, specialized LLMs could provide a more secure alternative to cloud-based services. These models could be fine-tuned specifically for database migration tasks while maintaining data sovereignty. Second, privacy-preserving techniques such as query anonymization and schema obfuscation could be integrated into the translation pipeline. Finally, access control mechanisms could be implemented to ensure that only authorized personnel can utilize the migration tools. The balance between leveraging LLM capabilities and maintaining data security presents an important challenge for future research. As these technologies mature, developing standardized security protocols for LLM-assisted database operations will become increasingly critical.

7. Conclusion

This paper introduces a novel LLM-powered framework for automated database query translation, a critical step in modernizing data systems. By combining in-context learning with iterative refinement, our approach effectively translates queries between different KG paradigms (SPARQL and Cypher) with high accuracy (89.6%), reliability (97.3% executable rate), and structural fidelity. We have demonstrated that this method is not only scalable to enterprise-level datasets but is also highly cost-effective, offering a practical path for organizations to overcome the significant barriers of traditional database migration.

The key innovation of our work lies in its ability to bridge the semantic and structural gaps between query languages without large parallel corpora or manually engineered rules. This capability is essential for enabling the adoption of modern database technologies, such as Neo4j, which are increasingly favored for building sophisticated applications in domains like smart cities and eHealth. Our framework empowers organizations to evolve their data infrastructure while preserving valuable investments in existing business logic and applications.

While the results are strong, future work can extend this framework. Exploring advanced hybrid retrieval methods, developing robust domain adaptation techniques for out-of-distribution KGs, and extending the approach to other query language pairs are promising research directions. Furthermore, addressing privacy and security concerns through on-premise models and data anonymization will be crucial for production deployments.

In conclusion, our research establishes a new paradigm for database evolution. It provides a foundation for more agile, automated, and accessible migration processes, enabling data systems to keep pace with the rapid advancements in technology and application demands.

CRediT authorship contribution statement

Shangqing Zhao: Writing – original draft, Software, Project administration, Investigation, Conceptualization, Visualization, Resources, Methodology, Data curation. **Qifan Zhang:** Visualization, Writing – review & editing, Validation. **Man Lan:** Supervision, Conceptualization, Writing – review & editing, Funding acquisition.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used claude and writefull to check grammar and expressions. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Funding

This work was supported by National Natural Science Foundation of China [grant numbers 72192820, 72192824]; Pudong New Area Science & Technology Development Fund, China [grant number PKX2021-R05].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, Zachary Ives, DBpedia: A nucleus for a web of open data, in: The Semantic Web, 2007, pp. 722–735.
[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, Jamie Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: Proc. of SIGMOD, 2008, p. 1247.
[3] Denny Vrandečić, Markus Krötzsch, Wikidata: A free collaborative knowledgebase, Commun. ACM (2014) 78–85.
[4] Siyu Duan, Yang Zhao, Knowledge graph analysis of artificial intelligence application research in nursing field based on visualization technology, Alex. Eng. J. 76 (2023) 651–667.
[5] Xiaotong Wu, Jiaquan Gao, Muhammad Bilal, Fei Dai, Xiaolong Xu, Lianyong Qi, Wanchun Dou, Federated learning-based private medical knowledge graph for epidemic surveillance in internet of things, Expert Syst. 42 (1) (2025) e13372.
[6] Philipp Seifer, Johannes Härtel, Martin Leinberger, Ralf Lämmel, Steffen Staab, Empirical study on the usage of graph query languages in open source java projects, in: Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, 2019, pp. 152–166.
[7] DB-Engines, DB-engines ranking: Graph DBMS, 2023.
[8] Lakshya A. Agrawal, Nikunj Singhal, Raghava Mutharaju, A SPARQL to cypher transpiler: Proposal and initial results, in: Proc. of KDD, 2022, pp. 312–313.
[9] Zihao Zhao, Xiaodong Ge, Zhihong Shen, Chuan Hu, Huajin Wang, S2CTrans: Building a bridge from SPARQL to cypher, in: International Conference on Database and Expert Systems Applications, 2023, pp. 424–430.
[10] Pavel Shvaiko, Jérôme Euzenat, Ontology matching: State of the art and future challenges, IEEE Trans. Knowl. Data Eng. 25 (1) (2013) 158–176.
[11] Hongyu Lei, Chunhua Li, Ke Zhou, Jianping Zhu, Kezhou Yan, Fen Xiao, Ming Xie, Jiang Wang, Shiyu Di, X-Stor: A cloud-native NoSQL database service with multi-model support, Proc. VLDB Endow. 17 (12) (2024) 4025–4037.

[12] Jianjun Chen, Rui Shi, Heng Chen, Li Zhang, Ruidong Li, Wei Ding, Liya Fan, Hao Wang, Mu Xiong, Yuxiang Chen, Benchao Dong, Kuankuan Guo, Yuanjin Lin, Xiao Liu, Haiyang Shi, Peipei Wang, Zikang Wang, Yemeng Yang, Junda Zhao, Dongyan Zhou, Zhikai Zuo, Yuming Liang, Krypton: Real-time serving and analytical SQL engine at ByteDance, Proc. VLDB Endow. 16 (12) (2023) 3528–3542.

[13] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, Luke Zettlemoyer, BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Proc. of ACL, 2020, pp. 7871–7880.

[14] Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, Hanwang Zhang, KQA Pro: A dataset with explicit compositional programs for complex question answering over knowledge base, in: Proc. of ACL, 2022, pp. 6101–6119.

[15] Yu Gu, Xiang Deng, Yu Su, Don't generate, discriminate: A proposal for grounding language models to real-world environments, in: Proc. of ACL, 2023, pp. 4928–4949.

[16] Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, Wenhu Chen, Few-shot in-context learning on knowledge base question answering, in: Proc. of ACL, 2023, pp. 6966–6980.

[17] Yantao Liu, Zixuan Li, Xiaolong Jin, Yucan Guo, Long Bai, Saiping Guan, Jiafeng Guo, Xueqi Cheng, An in-context schema understanding method for knowledge base question answering, in: Proc. of KSEM, 2024, pp. 419–434.

[18] Jorge Pérez, Marcelo Arenas, Claudio Gutierrez, Semantics and complexity of SPARQL, ACM Trans. Database Syst. (2009) 1–45.

[19] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, Yu Su, Beyond I.I.D.: Three levels of generalization for question answering on knowledge bases, in: Proc. of WWW, 2021, pp. 3477–3488.

[20] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, Andrés Taylor, Cypher: An evolving query language for property graphs, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 1433–1445.

[21] Victor Zhong, Caiming Xiong, Richard Socher, Seq2SQL: Generating structured queries from natural language using reinforcement learning, 2017.

[22] Kuan Xu, Yongbo Wang, Yongliang Wang, Zihao Wang, Zujie Wen, Yang Dong, SeaD: End-to-end text-to-SQL generation with schema-aware denoising, in: Proc. of ACL Findings, 2022, pp. 1845–1853.

[23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei, Language models are few-shot learners, in: Proc. of NeurIPS, 2020, pp. 1877–1901.

[24] Yu Gu, Vardaan Pahuja, Gong Cheng, Yu Su, Knowledge base question answering: A semantic parsing perspective, 2022.

[25] Stephen Robertson, Hugo Zaragoza, et al., The probabilistic relevance framework: BM25 and beyond, Found. Trends® Inf. Retr. (2009) 333–389.

[26] Jacob Devlin, Ming Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 2019, pp. 4171–4186.

[27] Nils Reimers, Iryna Gurevych, Making monolingual sentence embeddings multilingual using knowledge distillation, in: Bonnie Webber, Trevor Cohn, Yulan He, Yang Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, Online, 2020, pp. 4512–4525.

[28] Debayan Banerjee, Sushil Awale, Ricardo Usbeck, Chris Biemann, DBLP-QuAD: A question answering dataset over the DBLP scholarly knowledge graph, in: BIR@ECIR, 2023, pp. 37–51.

[29] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[30] Mateusz Pawlik, Nikolaus Augsten, Efficient computation of the tree edit distance, ACM Trans. Database Syst. 40 (1) (2015) 1–40.

[31] Mateusz Pawlik, Nikolaus Augsten, Tree edit distance: Robust and memory-efficient, Inf. Syst. 56 (2016) 157–173.

[32] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, Luke Zettlemoyer, Rethinking the role of demonstrations: What makes in-context learning work? in: Proc. of EMNLP, 2022, pp. 11048–11064.

[33] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, Weizhu Chen, What makes good in-context examples for GPT-3?, 2021.