LLM エージェントの KG 探索結果の活用による KGQA の効率化*

Utilizing LLM Agent's Exploration Results for Efficient KGQA

蓬田 綾香 Ayaka YOMOGIDA
三谷 陽 Akira MITANI

This paper proposes a method to enhance the efficiency of Knowledge Graph Question Answering (KGQA) by utilizing Large Language Model (LLM) agent exploration results. We convert the subgraphs obtained through exploration into SPARQL queries, using question-query pairs as few-shot Text-to-SPARQL learning examples. Experiments show improved SPARQL query conversion and answer accuracy compared to zero-shot methods. We achieve a 6x reduction in information retrieval time compared to full agent-based exploration, with a slight decrease in answer accuracy. This efficiency makes our method suitable for time-sensitive applications and contributes to more efficient KGQA systems.

Keywords :
Natural Language Processing, Large Language Model, Knowledge Graph, Question-Answering

1. はじめに

大規模言語モデル（Large Language Model：LLM）の発展により，様々な自然言語処理タスクにおいて高い性能が報告されており，LLM の適用範囲は日々広がっている．LLM はウェブ上の大規模コーパスで学習されており，一般的な常識に加えて数学や法律などの専門的な知識も有している．一方で，学習データに含まれていない知識への対応は不十分であり，精度低下やハルシネーションが懸念される．また，リアルタイムに知識を更新することが難しいという課題もある．

この問題に対処するため，外部の知識リソースとして知識グラフ（Knowledge Graph：KG）と LLM との連携が検討されている．KG は，エンティティ間の関係（リレーション）をグラフ構造で表した知識データベースであり，高い信頼性と広範な知識源として活用されている．また，データベースであるため知識の更新も容易である．このように KG の知識と LLM を組み合わせることで，LLM のさらなる性能向上が期待される．しかし，膨大な KG の情報をプロンプトにすべて組み込むことは困難であるため，タスクに応じた必要な知識を効率的に抽出する必要がある．

KG に対する質問応答（Knowledge Graph Question Answering：KGQA）では，自然文の質問を SPARQL などの検索用クエリに変換し（Text-to-SPARQL），回答となる知識を得る手法が研究されている．LLM の文脈内学習（In-context Learning：ICL）[1] を活用した Text-to-SPARQL 手法も提案されているが，識別子などの KG の情報を与えずに変換することは困難であり [2]，効果的な KG 情報の付与が課題である．近年，LLM の高度な推論能力に着目し，LLM をエージェントとして KG を探索することで，回答に必要な知識を抽出する手法も提案されている [3]．このアプローチは柔軟な情報抽出が可能である反面，逐次的な LLM の判断が必要となるため，処理時間が長くなることが課題である．

そこで本研究では，回答に必要な情報にたどり着くまでにエージェントが取得したエンティティおよびリレーションの探索経路のサブグラフを用いて，KGQA を効率化する方法を提案する．具体的には，探索経路のサブグラフを SAPRQL クエリに変換し，質問文とのペアを Text-to-SPARQL の few-shot learning の事例として用いる．加えて，サブグラフ中のエンティティおよびリレーションの識別子情報もプロンプトに与えることで，新規の質問文を直接 SPARQL クエリに変換し，素早く KG から情報を抽出する．

*（一社）人工知能学会の了解を得て，「人工知能学会全国大会論文集」，2025，JSAI2025 巻，第 39 回（2025），セッション ID 1F4-OS-40b，p. 1F4-OS-40b-04 より一部修正して転載

2. 関連研究

1. でも述べたように，近年では LLM の ICL を活用し，個別のデータに fine-tuning することなくプロンプトベースで KGQA を扱う手法が提案されている．Text-to-SPARQL 手法において，Kovriguina らは，質問文，その質問に関連する最小のサブグラフ，Text-to-SPARQL の変換例の one-shot の以上 3 点をプロンプトに与える付与することで，LLM による自然文から SPARQL クエリへの変換精度が改善されることを報告した [4]．Avila らは，事前に KG のクラス，リレーションおよびエンティティ情報をインデックス化し，質問文に含まれる単語からインデックスを検索して得られたサブグラフをプロンプトに与え，SPARQL クエリに変換する手法を提案した [5]．ICL による Text-to-SPARQL 手法は，エンティティなどの識別子の接地が困難であると江上らも報告しているように [2]，プロンプトへの識別子情報の付与が課題である．

LLM の推論能力に着目した手法では，KG を環境，LLM をエージェントとし，質問の回答に関連するエンティティとリレーションを探索しながら必要な知識を取得して回答を行う．Jiang らは LLM と KG の連携のため，エンティティが有するリレーション抽出，ヘッドエンティティとリレーションに紐づくトリプル抽出の 2 つのインターフェイスを提案した [3]．Sun らはエンティティおよびリレーションの探索をビームサーチで反復できる手法を提案した [6]．Xiong らは KG を探索した情報を順次 SPARQL クエリに変換するといった探索と Text-to-SPARQL の中間的な手法を提案している [7]．これら探索を用いる手法は fine-tuning したモデルにせまる精度が報告されているが，逐次的に判断するアプローチのため，LLM の呼び出し回数が増大し，処理時間が長くなることが課題である．

3. LLM エージェント探索結果の活用

本研究では，質問文とその探索経路のサブグラフを表す SPARQL クエリの組み合わせデータ，およびサブグラフ中のエンティティおよびリレーションの識別子情報を活用する．このデータを few-shot learning の事例としてプロンプトに与えることで，新規の質問文を SPARQL クエリに変換して回答に必要な知識を取得する．これにより，従来の KG 探索を用いる手法と比較して，LLM の呼び出し回数を大幅に削減し，KGQA の効率化を実現する．Fig. 1 に全体のフローを示す．

3.1 LLM エージェントによる KG の探索

先行研究に基づき，KG の探索は以下の 5 つのステップで構成した．

質問文のトピック抽出：はじめに，与えられた質問文から LLM を用いてトピックを抽出する

エンティティリンキング：抽出したトピックを用いて，探索の起点となる KG 上のエンティティを検索し，ヘッドエンティティ h とする．

シングルホップのトリプル取得：h を起点とするシングルホップのトリプル $(h, r_i, t_{i,j})$ を取得する．ここで $t_{i,j}$ は h および r_i に紐づく j 番目のテールエンティティを指す．

トリプルの選択：取得したトリプルのうち，回答を得るために必要と考えられる経路を LLM に選択させる．この時，経路は複数選択を可能とした．

情報の判断：現時点までに探索した経路で回答のため

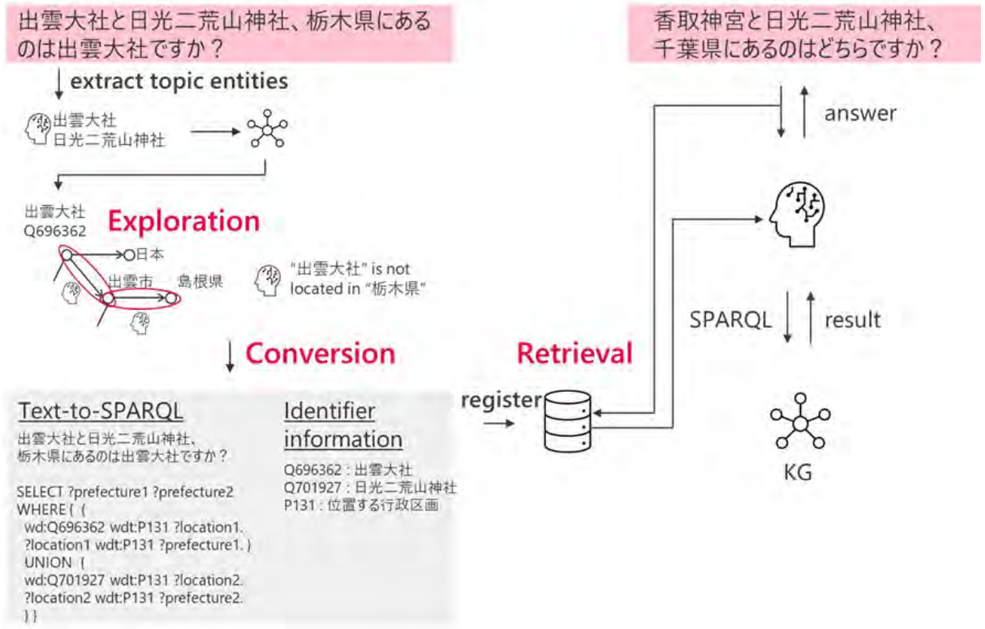

Fig. 1 The overview of the proposed approach

に十分な情報であるかを LLM に判断させ，十分でないと判断された場合，選択したトリプルのテールエンティティをヘッドエンティティとして探索を続行し，十分であると判断されるまで探索を繰り返す．

3.2 探索で得られたサブグラフの SPARQL クエリへの変換

探索により得られたエンティティおよびリレーションのサブグラフを LLM に与え，SPARQL クエリへの変換を行う．探索を開始したエンティティ以外は変数に置き換えることで，質問文の状態から変換可能な Text-to-SPARQL のデータとして保存する．

3.3 few-shot learning による SPARQL クエリ変換

新規の質問文に対し，過去探索した質問文との類似度を計算し，3.2 で変換したデータを few-shot learning の事例として用いる．質問文を SPARQL クエリに変換する指示をするとともに，類似度上位 k 件の選択されたデータをプロンプトに付与する．変換された SPARQL クエリを実行し，実行結果を基に質問に対する回答を LLM で行う．

4. 実験

日本語を対象に実験するため，JEMHopQA[8] を用いた．本データセットは Wikipedia のページタイトル，冒頭部分などのハイパーリンクを対象に作成した知識に基づく QA データセットである．

そのため，Wikidata や森羅 [9] といった既存の KG に保存されていない知識を必要とする QA も含まれている [10]．今回は Wikidata を探索対象とし，JEMHopQA に対して Wikidata で回答可能な QA を選定し，回答可能な train データの探索結果を SPARQL クエリへ変換した．valid データは新規の質問文として扱い，類似する train データに紐づく変換結果を付与し，SPARQL クエリ実行結果精度および質問応答の回答精度を評価した．本実験では，Azure OpenAI Service の API サービスの gpt-4 モデルを使用した．

4.1 Wikidata で回答可能なデータの選定

今回は日本語ラベルを有するエンティティを対象としたため，数値比較を対象とする問題はあらかじめ除外した．JEMHopQA にはブリッジエンティティによって接続されるトリプル（$(h_1, r_1, t_1),(h_2, r_2, t_2), t_1 = h_2$）を要する構成質問と，同じリレーションを持つトリプル（$(h_1, r_1, t_1), (h_2, r_2, t_2), r_1 = r_2$）を要する比較質問の 2 種類が含まれている．比較質問において，2 つのトピック抽出後に元の質問文をそのまま探索の指針として用いると，経路選択や現時点の情報で回答可能か判断する推論に影響を及ぼしたため，

以下の例のように，トピックおよびトピックごとの質問文を LLM に生成させた．

質問文：出雲大社と日光二荒山神社，栃木県にあるのは出雲大社ですか？

・トピック：出雲大社

・分割後質問文：出雲大社は栃木県にありますか？

・トピック：日光二荒山神社

・分割後質問文：日光二荒山神社は栃木県にありますか？

抽出したトピックを用いて，Wikidata API で取得したエンティティ名と識別子の候補を LLM に与え，探索を開始するエンティティを判断させた．エンティティに紐づくトリプルは Wikidata クエリサービスを使用して取得した．取得したトリプルは文字数削減のため，エンティティおよびリレーションのラベルを用いて（h, r, t）に変換してからプロンプトに与え，回答に関係する経路を選択させた．回答可能なデータの判定基準は，リレーションおよびエンティティが JEMHopQA の正解と完全に一致してなくても，回答を推論するために適切なトリプルである場合は回答可能とした．探索を行った結果，train データで 236/1,059 件（22.3%），valid データで 20/120 件（16.7%）回答可能であった．

4.2 train データの SPARQL クエリ変換結果

4.1 で選定した train データに対し，SPARQL クエリへの変換を行った．使用したプロンプトを Fig. 2 に示す．また，プロンプトを用いて変換した例を Table 1 に示す．ほとんどのデータで問題なく変換できたが，同一リレーションで複数の経路を選択していた場合，1 つのリレーションに対する SPARQL クエリに変換すれば所望の知識を取得できるが，経路の数だけ SPARQL クエリに変換される傾向が見られた．クエリの実行には問題ないが，手動で修正を行った．

```
あなたはグラフDBを扱う賢いアシスタントです。
あるノードを起点として質問に対する回答についての情報を得るための
グラフの経路を与えるので、SPARQLの検索式に変換してください。

{few-shot}

###
質問：{question}
起点：{start_node}
経路：{route}
検索式：
```

Fig. 2 Prompt for converting exploration results

4.3 valid データの評価結果

4.1 で選定した valid データ 20 件の質問文に対して SPARQL クエリへの変換を行い，変換したクエリの実行結果を用いた質問応答を行った．クエリ変換時には 4.2 のデータの内，質問文同士の類似度上位 10 件をプロンプトに与えた．類似度の計算には，Azure OpenAI Service の text-embedding-ada-002 を使用して得られたベクトルの cosine 類似度を使用した．shot として付与するデータには，探索に使用した特定のエンティティの識別子も含まれている．しかし，valid データで使

Table 1 Examples of question, exploration results and SPARQL conversion

question	exploration result	SPARQL

佐藤浩市さんのお父さんの血液型は何型ですか？	start_node : 佐藤浩市:Q748582,
	route:(佐藤浩市, 父, 三國連太郎),
	(三國連太郎, 血液型,AB型),
	父:P22, 血液型:P1853
		SELECT ?father ?bloodType WHERE {
		wd:Q748582 wdt:P22 ?father .
		?father wdt:P1853 ?bloodType .
		}

ダークナイト』と『ジュラシック・ワールド』、クリストファー・ノーランが監督してるのはどちら？	start_node : ダークナイト:Q163872, ジュラシック・ワールド:Q3512046,
	route:(ダークナイト, 監督, クリストファー・ノーラン),
	(ジュラシック・ワールド, 監督, コリン・トレヴォロウ),
	監督:P57
		SELECT ?director1 ?director2
		WHERE {
		{ wd:Q163872 wdt:P57 ?director1. }
		UNION
		{ wd:Q3512046 wdt:P57 ?director2. }
		}

用されているエンティティは train データのエンティティですべてカバーされておらず，ハルシネーションが生じることが分かった．そのため，探索時と同様に候補となるエンティティ名および識別子を Wikidata API で取得してプロンプトに与えた．SPARQL クエリに変換するプロンプト例を Fig. 3 に示す．変換された SPARQL クエリを実行して回答に必要な正しいトリプルが得られたか，質問に対する回答が正しいかをそれぞれ Accuracy で評価した．比較のため，Wikidata API で取得した候補エンティティ名および識別子のみプロンプトに与えた zero-shot の精度も評価した．Table 2 に Accuracy の比較を示す．zero-shot と比較して提案手法は適切な SPARQL クエリに変換されており，それに伴い質問応答精度も向上した．zero-shot でも正しいトリプルを取得できた質問はあり，SPARQL クエリの基本的な構成および一部のリレーション名および識別子を gpt-4 は知識として有していることが分かった．2 つの手法を比較した場合，特に比較問題で変換される SPARQL クエリに差が見られた．それぞれ変換された SPARQL クエリの例を Table 3 に示す．zero-shot では，特定の配給元と一致するか直接確認するクエリに変換されており，質問文の表現をそのままクエリに変換する傾向が見られた．一方，few-shot ではそれぞれの映画の配給元を取得するクエリに変換されており，質問文に回答するために，より簡易なタスクに分解されたと考えられる．探索結果を付与することで，質問文の表現によらず回答に必要な情報を取得できる簡易なクエリに変換できることが示唆された．

提案手法の few-shot で正しいトリプルを取得できなかった質問は次の 2 パターンであった．

1. 適切な事例が train データに存在しない
2. 適切な事例を付与できているが，適切なクエリに変換できていない

2. では，ある特定の場所が所在する県に関する質問であった．県の情報を取得するためには，ある場所の “ 位置する行政区画 ” で市の情報を取得し，さらに市の “ 位置する行政区画 ” を取得する必要がある．しかし，今回変換した結果では 2 段階目の市の “ 位置する行政区画 ” を取得する SPARQL クエリが得られていなかった．与えた類似事例の中に，市の所在する県を尋ねる質問が含まれており，市と特定の場所で県にたどり着くまでに必要な SPARQL クエリとの違いを few-shot で学習しきれていないことが要因と考えられる．今回のプロンプトでは SPARQL クエリのみ出力させたが，どのようなクエリが必要か段階的に推論させるといった対応が考えられる．

Table 2 Comparison of accuracy across methods

method	Accuracy of Correct Triple Retrieval	Answer Accuracy
few-shot(ours)	0.80	0.75
zero-shot	0.50	0.45
Agent-based exploration	0.90	1.00†

† The reason for the higher answer accuracy is that the correct response could be generated by supplementing the internal knowledge of the LLM. We have confirmed that the supplemented knowledge exists in Wikidata and that it is possible to retrieve the information and answer correctly using an appropriate SPARQL query.

```
あなたはグラフDBを扱う賢いアシスタントです。
ユーザの質問に対し、回答するために必要な情報をグラフDBから取得するSPARQLの検索式を生成してください。
与えられたidから検索式の生成に必要なidを1つ選択してください。

trainデータ類似度上位10件
{10-shot}

###
質問：{question}
id：{suggested_idx}
検索式：
```

Fig. 3 Prompt for text-to-SPARQL

Table 3 Examples of converted SPARQL queries

few-shot(ours)	zero-shot

SELECT ?distributor1 ?distributor2
WHERE {
{ wd:Q189505 wdt:P750 ?distributor1. }
UNION
{ wd:Q83965418 wdt:P750 ?distributor2. }
}
	SELECT ?film ?filmLabel
	WHERE { VALUES ?film {wd:Q189505 wd:Q83965418}
	?film wdt:P750 wd:Q154950.
	SERVICE wikibase:label { bd:serviceParam
	wikibase:language "[AUTO_LANGUAGE], en". } }

4.4 KG からの情報取得時間比較

4.1 でのエージェントによる探索および本手法による KG からの情報抽出時間の平均を Table 4 に示す．質問文を直接クエリに変換することで LLM の呼び出し回数を低減でき，情報抽出時間を 1/6 に短縮できた．迅速な対応が求められる実用的なアプリケーションにおいて有効であると考えられる．4.3 でも述べたように，エージェントによる探索では回答可能な質問に対し，適切な事例が付与できない場合では SPARQL クエリへの変換が困難なケースもある．参照可能なデータ数や類似スコアの状況に応じてこれらの手法を使い分けることで，両者の利点を最大限に活かすことができると考えられる．

Table 4 Comparison of information retrieval time from KG (sec)

Agent-based exploration	few-shot(ours)
59.3	11.0

5. おわりに

本研究では，LLM エージェントの KG 探索経路のサブグラフを Text-to-SPARQL および識別子の情報として活用し，KGQA の効率化を試みた．提案手法は zero-shot と比較して SPARQL クエリ実行結果の適切性および質問応答の回答精度を向上させることを示した．また，エージェントによる探索手法と比較した場合，回答精度は低下するものの，KG からの情報取得時間を 1/6 に短縮できた．今後の取り組みとして，より複雑なクエリ変換に対する効果検証のためのデータセットの拡張，実用的なアプリケーションにおける探索とクエリ変換の組み合わせ方法の検討を行っていく．

参考文献

1) Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever and Dario Amodei. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems 33, (2020).
2) 江上周作，福田賢一郎．大規模言語モデルを用いた SPARQL クエリ生成の予備的実験．第 60 回人工知能学会セマンティックウェブとオントロジー研究会，(2023).
3) Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao and Ji-Rong Wen. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, (2023).
4) Liubov Kovriguina, Roman Teucher, Daniil Radyush and Dmitry Mouromtsev. SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation. Proceedings the 19th International Conference on Semantic Systems, (2023).
5) Caio Viktor S. Avila, Vania M.P.Vidal, Wellington Franco and Marco A. Casanova. Experiments with text-to-SPARQL based on ChatGPT. 2024 IEEE 18th International Conference on Semantic Computing (ICSC), (2024).
6) Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum and Jian Guo. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. The Twelfth International Conference on Learning Representations, (2024).
7) Guanming Xiong, Junwei Bao and Wen Zhao. Interactive-KBQA: Multi-Turn Interactions for Knowledge Base Question Answering with Large Language Models. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, (2024).
8) 石井愛，井之上直也，鈴木久美，関根聡．JEMHopQA：日本語マルチホップ QA データセットの改良，言語処理学会第 29 回年次大会発表論文集，(2024).
9) 関根聡，安藤まや，小林暁雄，隅田飛鳥．拡張固有表現定義の更新と日本語 Wikipedia 分類データ 2019．言語処理学会第 26 回年次大会発表論文集，(2020).
10) 石井愛，井之上直也，鈴木久美，関根聡．マルチホップ QA の根拠情報を用いた LLM の “ 偽 ” 正解の分析．言語処理学会第 29 回年次大会発表論文集，(2024).

著者

蓬田 綾香
よもぎだ あやか

AI 研究部
自然言語処理や AI 技術の開発に従事

三谷 陽
みたに あきら

AI 研究部　博士 (理学)
データ分析や AI 技術の開発に従事