言語モデルの外部知識獲得に関する高速化と小説プロットによる性能評価

近藤 碧　　小田 幹雄
久留米工業高等専門学校
k-104_2024@kurume.kosen-ac.jp　oda@kurume-nct.ac.jp

概要

大規模言語モデル（LLM）をサービスに組み込むとき，外部知識の獲得および推論速度，推論精度は重要である．そこで，本研究では，LLM の推論速度を向上させるために，Hybrid-RAG を改善した外部知識の獲得手法を提案する．従来手法においては，グラフデータベースとベクトルデータベースを並列に接続していたが，提案手法では，各々を直列に接続することにより効率的な外部知識検索を実現した．また，実験において，小説プロットを外部知識として導入したときの性能評価を行った．その結果，従来手法と比較して推論速度が向上することがわかった．

1 はじめに

近年，大規模言語モデル (Large Language Model, LLM) により，一つの言語モデルで汎用的に文章を生成することが可能になった．現在，LLM 技術で最も利用されている OpenAI 社の ChatGPT[1)] において，GPT-4 モデルでは，GPT-3 モデルと比較してより多くのタスクを解くことができる．このように，LLM を用いることにより様々なタスクを解くことができるようになった．

LLM は，ChatGPT のほかに，Google 社が開発している Gemini[2)]や，Anthropic 社が開発している Claude[3)]などがある．これらのモデルは，Web インタフェースおよび API の形式で提供され，API を使用すれば，開発者が自由に開発するアプリケーションに組み込むことが可能である．このように，近年，高性能な LLM を容易にサービスへ搭載することが可能になっている．しかしながら，このようなサービスを運用するとき，多くの場合に LLM が学習段階で獲得していない知識を利用する必要があり，事前学習データのみで構築された LLM においては，システム実装において困難が生じる．そこで，外部知識を獲得するための手法として，例えば，Retrieval Augmented Generation(RAG)[?] と Long-Context[?] がある．

LLM を用いてサービスを運用する場合，推論時の精度および速度が重要である．精度および速度は，それぞれ，サービスの品質およびユーザビリティに直接影響する要素である．そこで，本研究は，高速な応答が求められるサービスを運用するために，RAG の一種である Hybrid-RAG[?] をベースに推論速度を改善する手法を提案する．実験では，サービスの一例として小説執筆の生成を行う LLM の運用を想定する．小説のプロットデータを外部知識として，プロットに関する質問を LLM に入力した際の提案手法と Long-Context の性能比較を行う．さらに，サービス運用時に必要な API の課金コストについて考察する．

2 関連研究

2.1 RAG

RAG は，データベース（Index），検索機構（Retriever），生成機構（Generator）からなり，LLM に外部知識を提供する技術である．具体的には，まず，外部知識を格納した Index を用意し，入力されたプロンプトに基づいて，Retriever が Index を検索し，関連する知識を取り出す．つぎに，取り出した外部知識をもとに，Generator がプロンプトに応じた適切な出力を生成する．RAG を用いることで，LLM が外部知識を参照でき，学習時に含まれていない知識も柔軟に利用することができる．RAG の手法は抽象化されており，Index で使用するデータベースを変更することで，検索精度および効率を改善でき

1） https://chatgpt.com/
2） https://gemini.google.com/
3） https://claude.ai/

る．RAG の Index として，ベクトルデータベースを用いる手法 [?] とグラフデータベースを用いる手法 [?] が提案されている．

ベクトルデータベースを用いる手法

外部知識を埋め込みベクトルとして表現したデータを格納するデータベースである．ベクトルデータベースを用いた RAG は，他の手法と比較して Retriever が高速であることが特徴である．プロンプトに入力された文章から類似度を計算し，類似度が高い外部知識をベクトルデータベースから引き出し，外部知識を獲得することができる．

この手法の欠点は，各テキストドキュメントの文章の関連性を保存することができないことである．これは，ベクトルデータベースが，テキストドキュメントの埋め込み表現のみを格納しているためである．

グラフデータベースを用いる手法

グラフ構造のデータを格納するデータベースである．グラフデータベースを用いた RAG は，グラフの関係性を用いて効率良くプロンプトの回答に必要な情報を検索することができる．LLM が，プロンプトに入力された文章から，グラフデータベースを操作するクエリを発行し，そのクエリを用いてプロンプトに応じた外部知識をグラフデータベースから引き出し，外部知識を獲得することができる．

この手法の欠点は，データベース上の外部知識を網羅的に検索することができないことである．これは，グラフデータベースがグラフ構造であるため，ノードの周辺しか探索を行うことができないためである．

2.2 Hybrid-RAG

ベクトルデータベースを用いる手法とグラフデータベースを用いる手法を組み合わせた手法として，Hybrid-RAG[?] がある．これは，**??**節のそれぞれの手法の欠点を補い，グラフデータベースに対して，ベクトルデータベースの機構を埋め込むものである．Hybrid-RAG では，ベクトルデータベースを利用して外部知識の全体構造を埋め込み表現に変換し，グラフデータベースを並列に構築することで，網羅的に検索を行いながらドキュメントに関係性を持たせることを実現している．

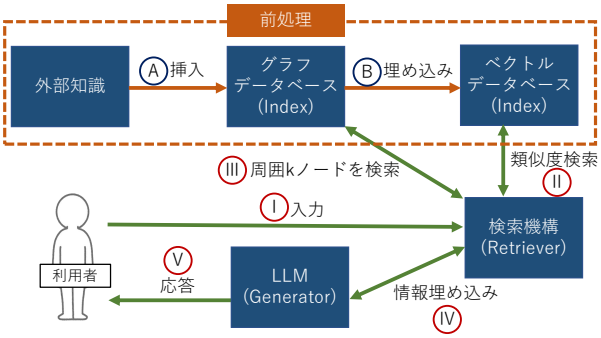

図 1 提案手法の概要

2.3 Long-Context

Long-Context は，Transformer モデルの Context Window の大きさに着目した手法である．Context Window とは，LLM が一度に処理することができるトークン数であり，例えば，Google 社の Gemini 1.5 Pro は，最大 200 万トークンの Context Window をもつ [?]．このような巨大なトークンを扱うことができるモデルの場合，RAG のように外部知識をデータベースを経由して参照する必要はなく，代わりに LLM に入力するプロンプト内にタスクを解くために必要な情報を組み込むことで同様の機能が実現できる．

RAG と Long-Context の性能差を比較した研究 [?] は，一般に，Long-Context の方が RAG より推論性能はよいと報告している．しかしながら，一般に，LLM は入力するトークン数が長くなるほど推論時間も長くなる．そのため，高速に推論を行いたい場合は，RAG が有効であると考えられる．本研究は推論速度の改善が目的であり，Long-Context ではなく Hybrid-RAG をベースとしたアプローチを提案する．

3 提案手法

提案手法は，前処理および推論処理に大別され，その概要を図**??**に示す．前処理では，LLM へ導入する外部知識をグラフデータベースとベクトルデータベースに格納する．推論処理では，利用者の入力に関連する外部知識を検索機構を通して獲得し，それを基に LLM に対して情報の埋め込みを行い，入力に最適な応答を生成する．

提案手法は，従来手法の Hybrid-RAG の改良モデルであり，グラフデータベースとベクトルデータベースを直列に接続していることが特徴である．従来の Hybrid-RAG は，外部知識であるドキュメントインデックスをグラフ構造および埋め込み表現にそれぞれ変換し，グラフデータベースおよびベクトル

データベースに格納するが，提案手法では，外部知識から作成したグラフの一部を埋め込み表現に変換し，ベクトルデータベースに格納する．これにより，検索機構における類似度検索の高速化が期待される．

つぎに，提案手法の詳細な処理手順を示す．手順 A および B は，前処理であり，手順 I から V は推論処理である．なお，前処理は，外部知識を更新するときに実行され，推論処理は利用者から入力が与えられたときに実行する．以下に，各手順の詳細を述べる．

手順 A．挿入

外部知識をグラフデータベースに挿入する．なお，各々のサービスに最適化することを想定しているため，グラフの定義法やデータベースの構成は限定しない．

手順 B．埋め込み

グラフデータベースに保存されているノードに関する説明を埋め込み表現として保存する．なお，埋め込み表現の手法については限定しない．

手順 I．入力

利用者が入力したプロンプトを，手順 B の埋め込み表現手法を用いてベクトル化する．

手順 II．類似度検索

埋め込み表現に変換されたプロンプトと類似度が高いノードの説明をコサイン類似度により検索する．

手順 III．周囲 k ノードを検索

手順 III における，最も類似度が高いノードからの距離が k 以下のノードを取得する．このとき，k はハイパーパラメータである．

手順 IV．情報埋め込み

手順 III において，取得されたノードに保存されている説明と利用者から入力されたプロンプトを結合し，LLM に入力する．

手順 V．応答

手順 IV で入力された結合情報により推論を行い，応答をユーザに返す．

4　実験

4.1　実験方法

実験では，外部知識として小説のプロットデータを使用し，プロットに関する質問を入力し，その回答を出力させる．グラフデータベースとして，Neo4j[4] を用い，手順 A における外部知識からグラフ構造の獲得は，例えばデータ**??**および付録のデータ**??**に示す XML 形式の情報から図**??**に示すグラフ構造に変換することにより行う．なお，データ**??**はデータ**??**の 20 行目以降の内容である．また，手順 B では，HuggingFace で公開されている cl-nagoya/ruri-small[5] を用いてグラフの埋め込み表現

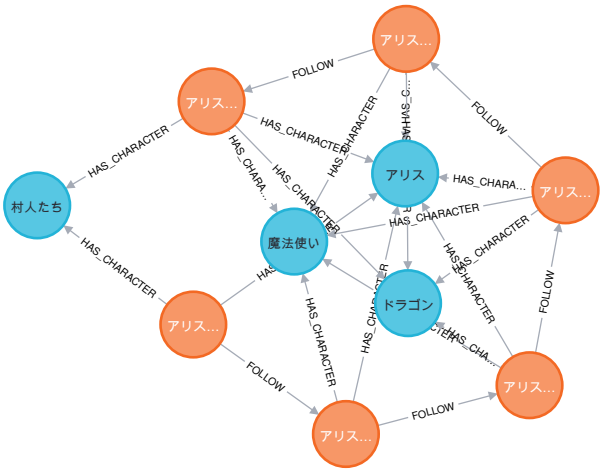

図 2　グラフ構造の一例

データ 1　入力に用いた XML の一部

```
1  <!-- 登場人物の情報を記述する-->
2  <Characters>
3      <Character id="1" role="主人公">
4          <Name>アリス</Name>
5          <Description>勇敢で賢い少女。</
               Description>
6      </Character>
7      <Character id="2" role="敵役">
8          <Name>ドラゴン</Name>
9          <Description>恐ろしい火を吐くド
               ラゴン。</Description>
10     </Character>
11     <Character id="3" role="脇役">
12         <Name>村人たち</Name>
13         <Description>アリスの住む村の住
               人たち。</Description>
14     </Character>
15     <Character id="4" role="ヒロイン">
16         <Name>魔法使い</Name>
17         <Description>アリスの冒険を助け
               る魔法使い。</Description>
18     </Character>
19 </Characters>
```

4）　https://neo4j.com/
5）　https://huggingface.co/cl-nagoya/ruri-small

表 1　実験結果 ($k = 1$)

	google/gemma-2-9b-it			llm-jp/llm-jp-3-13b-instruct		
	トークン数 [token]	推論時間 [s]	正答率 [%]	トークン数 [token]	推論時間 [s]	正答率 [%]
Long-Context	866	1.928	60	876	23.44	**100**
Hybrid-RAG	379	1.351	60	353	15.65	80
提案手法	**163**	**1.198**	**80**	**155**	**12.66**	80

を獲得 [**?**] する．また，LLM として，google/gemma-2-9b-it[6]，llm-jp/llm-jp-3-13b-instruct[7] を用い，それぞれのモデルごとに性能評価を行う．さらに，従来の Hybrid-RAG および Long-Context を用いたモデルとの比較実験も行う．Long-Context では，グラフ構造のデータを文章で表すために，XML 形式のデータを入力として使用し，データ**??**およびデータ**??**を用いる．

4.2　実験結果

外部知識に関する質問 5 種を提案手法，Hybrid-RAG および Long-Context に入力したときに使用されたトークン数および推論時間の平均ならびに正答率を表**??**に示す．提案手法のトークン数および推論時間について，各々のモデルで性能が向上したことがわかった．また，一部のモデルでは提案手法の正答率が向上することがわかった．

5　考察

実験結果より，提案手法は従来手法である Hybrid-RAG および Long-Context を用いた手法よりも高速なアルゴリズムであることがいえる．また，正答率は，google/gemma-2-9b-it を用いた場合に Hybrid-RAG および Long-Context を用いた手法より改善することがわかった．llm-jp/llm-jp-3-13b-instruct を用いた場合は，Long-Context を用いた手法よりも正答率が下がるが，Hybrid-RAG と正答率は同一であった．これより，提案手法は，Hybrid-RAG および Long-Context よりも高速であり，精度については，手法により性能差があるが，概ね Hybrid-RAG と同一かそれよりもよいといえる．

本研究の実験においては，推論速度とともに，LLM を用いたサービスを運用するときにコスト面で重要なトークン数の算出も行った．近年の LLM は，API を使用する場合，利用するトークン数に応じた料金を支払う必要がある．本稿執筆時点の主要な LLM サービスにおいて，Long-Context および Hybrid-RAG，提案手法を用いた手法において料金面でどのような違いがあるか比較を行った結果を表**??**に示した．なお，Long-Context，Hybrid-RAG および提案手法の料金の計算には，各々の実験の平均トークン数である 871 トークン，366 トークン，159 トークンを用いた．結果より，提案手法を利用する方が従来手法を使用するよりも料金が安いといえる．これより，提案手法が LLM を用いてサービスを運用する場合において有効であることが確認できた．

表 2　トークン数と API の料金の関係 (単位:$)

	GPT-4o[8]	Gemini 1.5 Flash[9]
1 トークン	2.5×10^{-6}	7.5×10^{-8}
Long-Context	2.2×10^{-3}	6.5×10^{-5}
Hybrid-RAG	9.2×10^{-4}	2.7×10^{-5}
提案手法	4.0×10^{-4}	1.2×10^{-5}

6　おわりに

本研究では，LLM を用いたサービスを運用するような限られた使用用途において，外部知識を参照するときの推論速度及び推論精度を改善する手法を提案した．実験として，小説のプロット情報を外部知識として入力し，提案手法と従来手法の推論精度および推論速度の比較を行った．実験結果より，提案手法は従来手法を用いて推論するよりも推論速度の面から優れていることがわかった．また，推論精度においても大きく低下することはないことが確認された．さらに，LLM を用いてサービスを運用するような場合に必要な API のコストについて，提案手法を利用した場合と従来手法を利用した場合の料金差の比較を行った．これにより，提案手法が LLM を用いてサービスを運用するような場合において有効であることがわかった．今後は，提案手法を用いて LLM を使用するサービスを構築し，実運用上の検証を行いたい．

6）　https://huggingface.co/google/gemma-2-9b-it
7）　https://huggingface.co/llm-jp/llm-jp-3-13b-instruct
8）　https://openai.com/api/pricing/
9）　https://ai.google.dev/pricing/

謝辞

本研究を遂行するにあたって，久留米工業高等専門学校制御情報工学科古賀裕章准教授に一部助言をいただいた．

参考文献

[1] Patrick Lewis, Ethan Perez, Aleksandara Piktus, F Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, M Lewis, Wen-Tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. **Neural Inf Process Syst**, Vol. abs/2005.11401, pp. 9459–9474, May 2020.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L Ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I Guyon, U Von Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, **Advances in Neural Information Processing Systems**, Vol. 30. Curran Associates, Inc., 2017.

[3] Bhaskarjit Sarmah, Benika Hall, Rohan Rao, Sunil Patel, Stefano Pasquali, and Dhagash Mehta. HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. **arXiv [cs.CL]**, August 2024.

[4] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-augmented generation with graphs (GraphRAG). **arXiv [cs.IR]**, December 2024.

[5] Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Retrieval augmented generation or long-context LLMs? a comprehensive study and hybrid approach. **arXiv [cs.CL]**, July 2024.

[6] Hayato Tsukagoshi and Ryohei Sasano. Ruri: Japanese general text embeddings. **arXiv [cs.CL]**, September 2024.

付録

データ 2　入力に用いたプロットのイベントデータ（入力に用いた XML の一部）

```
<!-- 小説のイベントを記述する-->
<EventList>
    <Event id="1">
        <Description>アリスは村を離れ、冒険に出る決意をする。</Description>
        <Relationship>
            <CharacterRef from="1" to="3" relation="friend"/>
            <CharacterRef from="3" to="1" relation="friend"/>
        </Relationship>
    </Event>
    <Event id="2">
        <Description>アリスは魔法使いと出会い、冒険の仲間になる。</Description>
        <Relationship>
            <CharacterRef from="1" to="4" relation="friend"/>
            <CharacterRef from="4" to="1" relation="friend"/>
        </Relationship>
    </Event>
    <Event id="3">
        <Description>アリスはドラゴンと対決し、村を救うために戦う。</Description>
        <Relationship>
            <CharacterRef from="1" to="2" relation="enemy"/>
            <CharacterRef from="2" to="1" relation="enemy"/>
            <CharacterRef from="1" to="4" relation="friend"/>
            <CharacterRef from="4" to="1" relation="friend"/>
        </Relationship>
    </Event>
    <Event id="4">
        <Description>アリスとドラゴンが友達になる</Description>
        <Relationship>
            <CharacterRef from="1" to="2" relation="friend"/>
            <CharacterRef from="2" to="1" relation="friend"/>
        </Relationship>
    </Event>
    <Event id="5">
        <Description>アリスはドラゴンと魔法使いを連れて村へ帰る</Description>
        <Relationship>
            <CharacterRef from="1" to="2" relation="friend"/>
            <CharacterRef from="1" to="4" relation="friend"/>
            <CharacterRef from="2" to="1" relation="friend"/>
            <CharacterRef from="4" to="1" relation="friend"/>
        </Relationship>
    </Event>
    <Event id="6">
        <Description>アリスは村の住人たちから盛大に祝福される。</Description>
        <Relationship>
            <CharacterRef from="1" to="3" relation="friend"/>
            <CharacterRef from="3" to="1" relation="friend"/>
        </Relationship>
    </Event>
</EventList>
```