Knowledge Graph Information Retrieval and Complex Question Answering

Oscar Cuellar [*1] Marc-André Allard [*1] Michell M. Payano [*1]

Abstract

Large language models (LLMs) have shown impressive abilities in generating human-like text in tasks such as question answering, machine translation, and text generation. However, they suffer from several problems, such as difficulty in recalling specific facts, hallucinations, difficulty in generalizing for specific domains, and a knowledge cutoff date. To address these issues, we created a system that incorporates external knowledge in an LLM's context window in order to answer complex general questions. We initially used GPT-3.5 turbo as the LLM for our tests, but then moved to GPT4 for our final results. We tested various approaches to determine the best one. First, we used RAG to fetch the information necessary to answer each question from a vector database containing Wikipedia articles about those subjects. We then used the Wikidata knowledge graph (KG) containing that same knowledge in a graph format. Finally, we used a REACT based approach to traverse the KG and fetch the correct information to feed to the LLM. By combining a ReACT based approach with GPT4, we achieved an improvement of around 10% over the best result included in the QALD10 dataset original paper (Usbeck et al., 2023). The code for this project is available on github[1].

1. Introduction

Large Language Models (LLM) have become more popular every day, especially after the launch of ChapGPT in 2022. These kinds of models that are trained on a large-scale corpus, have shown great performance in various natural language processing (NLP) tasks, such as question answering for example (Pan et al., 2023). Despite their great success and popularity, they suffer from some limitations such as not being able to recall facts and from hallucinations, which make them not able to generalize well on specific domains or on new knowledge.

A potential solution for these issues is to incorporate external knowledge from external sources, which could be in the form of Knowledge Graphs (KGs) or vector representations. Specifically a KG is method to store and structure information from diverse sources, such as encyclopedias and databases, in the way of triples composed by head entity, relation and tail entity. Many authors such as (Zheng et al., 2018), (Talmor & Berant, 2018), (Miller et al., 2016) and others, have applied KGs successfully in QA applications and for generating structured queries. However, one major drawback of KGs is that they could be difficult to construct and can be incomplete.

Another proposed promising solution for incorporating external knowledge to a LLM is through the method called Retrieval Augmented Generation (RAG). As some authors confirm, (Lewis et al., 2020) and (Gao et al., 2023), RAG techniques have remarkable advantages and particularly have achieved state-of-the-art performance in many natural language processing tasks and allows for continuous knowledge updates and integration of domain specific information.

Overall, the goal of this project is to examine how a LLM can answer common knowledge questions by injecting it with external knowledge, from either a vector database and a KG. Specifically, we test the following approaches:

1. Injecting knowledge into the LLM using RAG over a vector database to answer questions. For this approach, we also compared the results given by the Nomic Embedding (Nussbaum et al., 2024), which is a reproducible and open source model, with the OpenAI text-embedding-3-small embedding.

2. Fetching all information directly related to the question's main entity in the KG and including it in the

[*]Equal contribution [1]Department of Computer Science and Operations Research, Montreal University, Montreal, Canada. Correspondence to: Oscar Cuellar <oscar.cuellar.nevares@umontreal.ca>, Marc-André Allard <marc-andre.allard.3@umontreal.ca>, Michell M. Payano <michell.mercedes.payano.perez@umontreal.ca>.

[1]https://github.com/marca116/IFT6759

LLM prompt.

3. Traversing the graph using ReAct (Yao et al., 2022), which is a technique that incorporates both reasoning and acting into a LLM.

After performing these experiments using GPT 3.5, we compare each result by using the LLM alone as a baseline. The best result was when using a ReACT based approach to traverse the KG, which increased the f1 score from 0.408 for our baseline to 0.518. We then further improved this score by using GPT 4 (0.627). As a last step, we linked our text answers back to an entity from the KG, which reduced our final score slightly (0.610). While this is a big improvement over the best result from the QALD10 paper (0.507), it is not the state of the art. The Observation-Driven Agent (ODA) framework (Sun et al., 2024) is a novel method that incorporates the ability to reason over a KG via a cyclical paradigm of observation, action, and reflection. This approach is similar to the ReACT based approach we used, and was tested on the same dataset as ours (QALD10). It showed significant improvement over the original paper's results, with a f1 score of 0.667.

2. Related work

Approaches for automatically extracting information from KGs range from generating structured queries as in Zheng et al. (2018) that builds graph querying templates and Talmor Berant (2018), that introduces ComplexWebQuestions (CWQ) and builds a retrieving language to perform QA over the KG. More recent approaches usually learn stored-knowledge and question-context representations, and pair them in order to retrieve relevant information. Miller et al. (2016) and Xu et al. (2019) employed a neural network architecture (Key-Value Memory Networks) which can work with either text or a KG. This allows models to use information from a KG when possible, and switch to unstructured text when no satisfactory answer can be found. Saxena et al. (2020) propose EmbedKGQA to solve Multi-hop QA over (possibly incomplete) Knowledge Graphs (KGQA), this works by embedding entities and relations in the graph and then embedding the question. A score selection module is then used to select the final answer by incorporating the question and relation similarity scores.

According to Lewis et al. (2020), RAG has emerged as a promising solution by incorporating knowledge from external databases (unstructured textual documents), in order to solve most of the challenges faced by LLMs related to hallucination and outdated knowledge, Gao et al. (2023), since it allows for continuous knowledge updates and integration of domain-specific information. Specifically, RAG involves an initial retrieval step where the LLMs query an external data source to obtain relevant information before proceeding to answer questions or generate text. Furthermore, Li et al. (2022) emphasize that RAG techniques have remarkable advantages and particularly have achieved state-of-the-art performance in many NLP tasks, such as dialogue, machine translation, language modeling, summarization, paraphrase generation and text style transfer. Pan et al. (2023) propose a characterization consisting of three general frameworks on how to integrate LLMs and KGs and simultaneously leverage their advantages, which they call KG-enhanced LLMs, LLM-augmented KGs and Synergized LLMs + KGs.

Some of the main problems for integrating KG and LLMs have to do with 1) extracting only relevant information and not overloading the reasoning capabilities of a LLM with redundant or noisy information, which may be due to the KG construction and/or retrieval techniques, and 2) feeding structured data to be interpretable by a LLM. Wen et al. (2023) proposes a simple method to feed the prompt of a conversational LLM with textually represented KG subgraphs. Promising approaches using Graph Neural Network (GNN) representations of KGs have emerged which aim to solve both problems, Yasunaga et al. (2021) incorporates a GNN attention mechanism to retrieve nodes and relations and then reason over them in an end-to-end manner, Yasunaga et al. (2022) uses self supervision to jointly train LLMs and KGs. Zhang et al. (2022) fuses LLM and GNN representations. Sun et al. (2021) incorporates a dynamic graph pruning mechanism and performs joint textual-graph reasoning over representations of both modalities.

In the LLM-guided reasoning realm, Press et al. (2022) builds upon the Chain of Thought Wei et al. (2022) paradigm and proposes Self-Ask as an approach to structure LLM reasoning by dividing a question into simpler self-constructed sub-questions, answering them and integrating them into a final answer. Another interesting and recent work about the integration of LLMs and KGs is the one conducted by Sun et al. 2024, where they propose a novel AI framework called Observation Driven Agent (ODA). The ODA framework incorporates KG reasoning abilities over global observation that enhances reasoning capabilities through a cyclical paradigm of observation, action and reflection.

3. Methodology

3.1. Data preprocessing

In order to use the Wikidata knowledge graph and create the vector database of Wikipedia articles, we first analyzed the SPARQL queries associated to each question and downloaded all the Wikidata entities that were used in order to get the answer (the context). We decided to download the questions entities in order to reduce the size of the search space when identifying the question's main entity. There are about

100 million items on Wikidata, which would introduce a lot of ambiguity otherwise.

Each entity was saved as a json file, which contains all the information directly linked to it in the knowledge graph. This include it's label, a list of aliases, a description, a list of direct properties and a list of external properties. The direct properties represents the relations this entity has with the outside and the external properties is the opposite. It should be noted that the Wikidata graph is non symmetrical, which means that the relations that goes from the entity to the outside are not the same as those from the outside coming in (there are usually a lot more of those). We use the words 'properties' and 'relations' interchangeably, since relations in the Wikidata KG are called 'properties'.

We then fetched the corresponding Wikipedia article for each of those entities. Then, we created a script to split each article into smaller chunks, and stored all of them in a vector database. Using this database, we are now able to do RAG over it to fetch the relevant information to answer questions using an LLM.

We used the Wikidata/Wikipedia API in order to do all this. This API is free, but is not always very use friendly and required a lot of different calls in order to gather all the information necessary. One issue is that the entity's relations with other entities are only specified by their ID (Q76, P141, Q15070048) instead of by their labels ("Barack Obama", "child", "Sasha Obama"). This means that all those labels needs to be downloaded in a separate call.

3.2. Querying the LLM

To answer questions using the LLM, we used the OpenAI API to query either the GPT 3.5 Turbo or GPT4 Turbo models. A certain amount of prompt engineering and post processing of the answers given by the LLM is necessary to obtain satisfactory results. The Chain of Thought technique is used to improve the quality of the answers by forcing the LLM to show it's reasoning for it's choices. We also used one shot learning, which consists in including an example of what a typical question and answer should look like. This helps the model answer in the correct format. Finally, we applied a certain amount of post processing to the answers to match the format present in the QA datasets. For example, all dates needs to be in a YYYY-MM-DD format, all yes or no questions need to be answered with booleans, etc.

3.3. Retrieval Augmented Generation

Retrieval Augmented Generation is a technique that extends the context of an QA or Information Retrieval model with relevant content, in order to help it solve a given task, huge amounts of information can be stored in a database and accessed when answering a question, in this case, enhancing the context of the LLM with factual information. RAG has been shown to generate more specific and grounded answers, preventing LLMs' hallucinations and yielding good results on standard QA benchmarks. To implement a RAG model, Wikipedia's vital articles are segmented into chunks of around 100-200 words, for each of those, a text encoder model is used to generate an embedding which is then indexed into a vector database. At test time, an embedding is generated for the question and used for retrieving chunks that are likely to contain useful information for answering it.

Given a question, RAG models usually retrieve chunks whose embedding have the highest inner product with the question embedding. In order to do that quickly, the database should be indexed for that specific operation. The FAISS library (Douze et al., 2024) implements an index with fast maximum inner product searches and is widely used for RAG applications. However, we found that it sometimes fails to retrieve all high scoring matches. Instead, we normalized all embeddings to unit L2 norm, and used an euclidean nearest neighbors index, which does retrieve all high scoring matches and behaves much better.

Three encoder models were tested for implementing RAG

- Following the Dense Passage Retrieval (DPR) model (Karpukhin et al., 2020), where two different BERT models (Devlin et al., 2019) are finetuned to maximize the inner product of the representation (embedding) of both the question and the chunk that contains its answer, we used a pretrained version of this model and found it doesn't perform well; for many questions it doesn't retrieve the most relevant chunks, apparently because the semantic content seems to add up. For example, given the question "What is the time zone of Salt Lake City?", the highest scored chunks mention timezone related content several times and don't mention Salt Lake City, whereas the correct chunk mentions timezone one time and Salt Lake City one time. This may be due to the treating of negative samples, the loss and the training sets used to fit the model in (Karpukhin et al., 2020). Authors report a high top 100 accuracy; feeding 100 noisy chunks as context to the LLM wouldn't perform well for this application.

- We tested the OpenAI embeddings model `text-embedding-3-small` to encode both questions and chunks. This model doesn't show the problems as the DPR model; a manual test shows that it usually retrieves the correct chunk in the top-1, and always does among the top-3.

- Similar to OpenAI, we tested the Nomic embeddings (Nussbaum et al., 2024), this model also finetunes

BERT with carefully designed tasks and data, yielding comparable performance to the OpenAI model.

3.3.1. RAG CONSIDERATIONS

For RAG, we found that using plain chunks extracted from articles is not always a good idea, as they sometimes lack the necessary context for 1) retrieving the correct passage and 2) extending the LLM context in an understandable manner. Techniques such as annotating chunks with article metadata (e.g. title and a description of its content) and using overlapping context windows, perform well to alleviate these problems. Extending accessible content for QA models with a potentially very big and multi-domain knowledge base is among the main advantages of RAG, in this scenarios, special care should be taken for addressing noisy information with the approaches mentioned. Also consider that storing and indexing big dimensional continuous representations could require high computing resources.

3.4. Answering questions using the KG

We first prompt the LLM to identify the question's subject entity (the entity from which we want to fetch information from). Then, we try to fetch the corresponding entity by doing an exact match between the identified entity name in the question and either the label or any of the aliases from the entities in the Wikidata entities. If a match is found, then all of that entities information is included in the LLM question and answer prompt (ex : the label, the description, the relations, etc.). Some amount of filtering is necessary to respect the context limits for the LLM (16k tokens for gpt3.5 turbo). Thus, all the properties including information that was found to be redundant or unnecessary to answer the questions was removed. This include a large number of 'id' properties (ex : Encyclopedia Britannica ID). Once the filtering is done, each remaining property is included in it's own message in the LLM context window in order to allow it to use this information during the call.

3.5. Answering questions using the KG (Wikidata directly)

As described previously, the Wikidata entities and wikipedia articles which are referenced in the questions were downloaded and used for the entity linking, either doing an exact match on the label or any of it's aliases. This was done to reduce the ambiguity when trying to match with an existing Wikidata item (there are more than 100M items on Wikidata). However, it's not a very realistic use case, since only a small subset of the entities is chosen from when doing the entity linking. In order to allow our system to answer any kind of questions, not only questions related to the small pool of previously downloaded Wikidata entities, we decided to query the Wikidata api at inference time. The principle remains the same, we try to match the entity identified by the LLM with a Wikidata entity by using either the label or the aliases. However, since we now match with a much larger pool of entities, we needed a way to disambiguate and chose the correct entity amongst the returned list.

To solve this issue, we used the LLM to disambiguate. We asked it to chose which entity returned by the query correspond with the given question's main entity. Once done, we fetched all of this entity's information on Wikidata and included it in a new LLM call, same as before.

3.6. Answering questions using a ReACT based Approach

There are serious limitations with the previous approach. For one, it can only answer questions 1 hop from the main entity. This is because the information we would need to give to the LLM will augment exponentially the further we get away from the main entity. Since some entities have a very large number of relations with the outside (ex: Germany), this information already barely fits inside GPT3 context window, which is limited to 16k tokens.

To solve this issue, we instead used a ReACT based approach to allow the LLM to reason over the KG and decide which relations to traverse in the KG in order to reach the correct answer. Since we can now give specific relations to the LLM as context, as opposed to all of them, this reduces the amount of tokens used by the model and allows us to move deeper into the KG without exceeding the LLM context window.

Similar to the ReACT technique, we repeat a 3 step process in order to traverse the graph. We first start with an 'action' step, listing all the main entities properties. Then, we move the the 'observation' step, where we ask the LLM which of these properties should be traversed in order reach the final answer. The final step is the 'thought' step, where we analyze whether the original question can now be answered with the information we extracted from the KG, or if we need to continue traversing the graph. If it's the former, we stop the process at the point, and give the chosen properties values as context to the LLM so the question can be answered. If it's the later, the process continue 1-hop deeper into the graph, starting from the child entities present on the other side of the previously chosen property. These entities now act as the current 'main' entity, from which we proceed to list their properties, decide which of these properties to traverse and finally answer the question using the extracted information as context. A diagram of this process is illustrated in Figure 1.

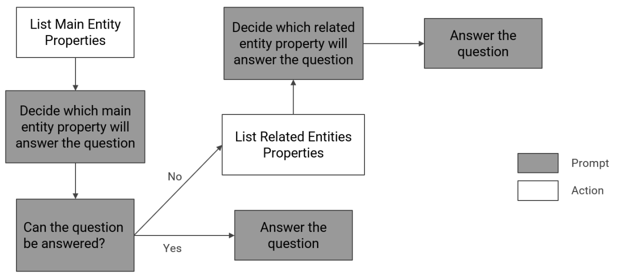
List Main Entity Properties
Decide which main entity property will answer the question
Can the question be answered?
No
Yes
List Related Entities Properties
Decide which related entity property will answer the question
Answer the question
Answer the question
Prompt
Action

Figure 1. KG traversal with ReACT

3.7. Applying entity linking to the answers

Because the datasets we used are intended to be used for question answering over linked data, the questions are usually answered with a link back to the corresponding entity in the KG. Since we answer the questions using an LLM, those answers are in a text format. In order to compare our results to those in the dataset, it's important to attempt to link our answers back to the corresponding entity in the KG. To do so, we used a similar approach as we did previously to determine the question's main entity. We queried the KG using the Wikidata API to find any entities that matched with our answers, and then asked the LLM to disambiguate if there was any ambiguity.

4. Experiments

4.1. Datasets

The QALD-10 dataset will serve as the primary dataset for this report. This dataset enables Knowledge Graph Question Answering (KGQA) systems to train and test over Wikidata (Usbeck et al., 2023). While this dataset supports questions in four languages, this project will concentrate solely on the English portion. Only a test set was created from scratch for this challenge, while the previous year's dataset (QALD-9-Plus (Perevalov et al., 2022)) was used as the training set. The QALD-10 benchmarking dataset, part of the QALD challenge series, solves many issues of other benchmarks such as poor translation quality for languages other than English. We will present our results for both the training and test datasets of QALD-9-Plus and QALD-10 (Table 3). However, for the sake of brevity, this report will focus specifically on the results from the QALD-10 test dataset.

4.2. LLM Baseline: No external knowledge

As a baseline, questions are fed as-is to the LLM, with no additional context or knowledge sources. The purpose is to compare this baseline with our other approaches, which does include external knowledge, and to see whether the results shows any improvement or not. We use GPT-3.5 turbo as the LLM (Ye et al.), given it's low cost. We also

QALD-9-plus dataset example	
Question	What is the time zone of Salt Lake City?
Context (Wikidata id)	Salt Lake City (Q23337), located in time zone (P421)
Answers (Wikidata id)	UTC07:00 (Q2212), Mountain Time Zone (Q3134980)

QALD-10 dataset example	
Question	On which stock exchanges are Siemens AG shares traded?
Context (Wikidata id)	Siemens (Q81230), stock exchange (P414)
Answers (Wikidata id)	New York Stock Exchange (Q13677), Frankfurt Stock Exchange (Q151139), Swiss Stock Exchange (Q661834), Xetra (Q819468)

Table 1. Dataset examples

use the default model parameters, except for the response length which is limited to 1750 tokens. We added this restriction to have better control over the token usage, and because we don't foresee any of the answers necessitating more than this. Comparing our results for the QALD10 test with those in table 2, we can see that this baseline fares pretty poorly, coming almost at last place (**0.408**).

4.3. Evaluation methods

To measure the performance of our results, we use the f1 score, since this is one of the most commonly used metrics to evaluate KGQA systems, according to an up-to-date leaderboard (A. Perevalov & Usbeck, 2022). The F1 score, which is the harmonic mean of precision and recall, tells us about the system's capacity to retrieve the right answer in terms of quality and quantity. As an example for a question in the QALD-9-plus dataset ”Who is the mayor of Berlin?”, if the true positive answered obtained is equal to the golden solved answer, which is ”Michael Müller”, then our f1 score is 1.

Table 3 contains our experiments results. We decided to include both our train and test sets results in that table. The reason is to be able to benchmark our system against not only the Qald10 test set, but also the QALD-9 plus train and test set (both papers results are included in table 2). It should be noted that the QALD-9 paper only include the paper's authors results due to limited external submissions, so that comparison should be taken with a grain of salt. Instead, we will mostly focus on the QALD10 test dataset results, of which 4 out of 5 systems are published.

4.4. RAG does not show improvement over baseline

We encountered issues with fetching the correct text passages while using the Dense Passage Retrieval (RAG) model, resulting in a score of **0.378**, which is significantly worse than the LLM baseline (**0.408**). OpenAI and Nomic embeddings on the other hand perform remarkably well for retrieval. However, performance of the QA model using these models is still not substantially different to the LLM-only baseline, with a score of **0.402** for the former and **0.411** for the later. This suggests that the high performing LLMs tested here already know the answers to common Wikipedia knowledge fairly well. Also, these benchmarks are designed to work over entities and their corresponding Knowledge Graph. We find two main challenges for a good performing RAG on this QA dataset:

- Questions refer to entities using explicitly the name with which they are found in the Knowledge Graph; directly searching using entity names will perform better than using a learned continuous text representation.
- Questions directly ask for a structured property or entity name, so using natural language as context for answering a question, instead of an entity-driven context format, can produce different answering styles than those in the benchmark, which further lower the performance of the model, for example, answering the question ”When was the De Beers company founded” with ”first half of 1888”, instead of plainly ”1888”.

Even though the RAG approach doesn't perform very well for answering questions here, it does retrieve the right text passages, this suggests that it can be used to retrieve KG entities in other scenarios where they are not addressed directly by their name.

QALD Papers Results

4.5. Significant improvements by including 1-Hop KG properties

We see a significant improvements when including all the information from the knowledge graph 1-hop away from the question's main entity (”Added KG Properties” row in Table 3). The F1 score increases from **0.408** for the baseline to **0.544**. It's important to note that the question's main entity is selected from the list of entities that were referenced in the original question, which removes most of the ambiguities in the entity linking part.

The accuracy of this entity linking is shown in Table 4 (Entity Linking Results). The number of entity that was matched with an existing Wikidata item is shown in the 'found' row, while the number where the entity linking failed is shown in the 'Missing' row. We can see that for the QALD10 (test) dataset, a corresponding entity was linked around **1/3** of the time (**295** found, **99** missing). However, since the LLM will still attempt to answer the question without external information in the cases where no entity is found, a failure in entity linking will not necessarily result in a failure in answering the question.

QALD10 Test

Author (System)	F1 Score
SPARQL-QA (M.A.B. Santana & Barbara, 2022)	0.507
Baseline/QAnswer (D. Diefenbach & Maret, 2017)	0.501
(K. Shivashankar & Steinmetz, 2022)	0.454
(N. Baramiia & Razzhigaev, 2022)	0.428
Singh & Gavrilev (no publication)	0.322

QALD9 Plus

Phase	F1 Score
Train	0.401
Test	0.446

Table 2. Original Paper Results for QALD10 and QALD9 Plus

4.6. Direct Wikidata queries yield similar results

We then tried using Wikidata directly instead of limiting ourselves to the previously downloaded Wikidata entities. Surprisingly, the results are still pretty good, with a F1 score of **0.498** on the QALD10 test set (see the 'added KG properties (Wikidata directly) row in Table 3). Looking at the results, we can see that the f1 score is a little lower than before, but it's still significantly higher than the LLM only baseline.

The number of instances where the entity linking is a success is actually higher than before when using Wikidata directly (**307** found vs **87** missing for QALD10 test). This is to be expected, since the pool of available entities is bigger than previously, which raises the changes of a match being found. However, there is no guarantee that the entity being linked is actually the correct one, which might partially explain the lower F1 score when compared to when using only the question's Wikidata entities.

4.7. More efficient KG traversal beyond 1-Hop using ReACT

We then tested our ReACT based approach to see how it compares to our previous ones. It seems there is a slight improvement over the previous result, going from an F1 score of **0.498** when including all the properties 1-hop away,

F1 score results (GPT 3.5)

	QALD10 (train)	QALD10 (test)	QALD9 Plus (train)	QALD9 Plus (test)
No external info (LLM only)	0.365	0.408	0.398	0.352
Dense Passage Retrieval (RAG)	0.352	0.378	0.382	0.331
OpenAI Emb (RAG)	0.340	0.402	0.347	0.356
Nomic Emb (RAG)	0.358	0.411	0.380	0.347
Added KG properties	0.502	0.544	0.511	0.470
Added KG properties (Wikidata directly)	0.471	0.498	0.498	0.461
ReACT	0.503	0.518	0.519	0.434

Table 3. GPT 3.5 F1 score results

Entity Linking Count

	QALD10 (train)	QALD10 (test)	QALD9 Plus (train)	QALD9 Plus (test)
Found	307	295	294	97
Missing	105	99	77	39
Total	412	394	371	136

Entity Linking Count (Wikidata directly)

	QALD10 (train)	QALD10 (test)	QALD9 Plus (train)	QALD9 Plus (test)
Found	336	307	309	102
Missing	76	87	62	34
Total	412	394	371	136

Table 4. Entity Linking Results

GPT 4

	QALD10 (test)
No external info (LLM only)	0.513
ReACT	0.627
ReACT + Answer Entity Linking	0.610

Table 5. GPT4 F1 score results

to **0.518** when using the LLM to traverse the graph (see 'ReACT in Table 3). This allows us to include entities into the LLM context window that are more than 1-hop away.

However, there are more benefits to this approach than the score itself would suggest. Limiting the information given to the LLM to only the properties used to traverse the graph reduces the overall token usage by about 40% (**1100** tokens per question on average vs **1900** before). This new approach is now also able to answer questions which would be impossible to answer before. For example, the question 'How many grand-children did Jacques Cousteau have?' requires 2-hops to answer, the 'child' relation needs to be traversed 2 times, once to reach his children and a second time to reach his grand-children.

4.8. GPT-4 outperforms GPT-3.5 significantly

We also tested the differences between using GPT3.5 and GPT4 as the LLM. Since GPT4 is around 20 times more expensive than GPT3.5, we decided to limit our tests to our baseline (LLM only) and our best approach (ReACT) on the Qald10 Test dataset. The F1 score for the goes from **0.408** when using GPT3.5 to **0.513** when using GPT4, and increase of more than 10% (see Table 5). A similar increase also occurs when comparing ReACT, with the F1 score going from **0.518** when using GPT3.5 to **0.627** when using GPT4, an increase of around 11%.

4.9. Linking answers to KG graph entities yield similar results

Finally, since the questions in the Qald10 dataset are all answered with a link back to the corresponding entity in the KG, we decided to do the same and attempt to link the text based answer given by the LLM back to an entity in the Wikidata KG. Even when faced with the ambiguity inherent to matching text with unique entities, the results are still very acceptable. The F1 score for the ReACT + GPT4 method goes from **0.627** to **0.610**, a decrease of around **1.5%** (See Table 5). This is significantly better than all the other results in the QALD10 paper, including the paper's authors own system, QAnswer, which has a score of **0.507** (See Table 2).

However, it should be noted that this result is not the state of the art, since an F1 score of **0.667** was achieved on this dataset two weeks before we submitted our own (Sun et al., 2024).

5. Conclusion

We tried multiple approaches to ground the LLM with external knowledge in order to improve over the LLM only baseline on the Qald10 benchmark. Including all properties of the question's main entity in the LLM context has resulted in significant improvement over the baseline (around **9%**). Using a React based approach to traverse the KG and include information located more than 1-hop away resulted in even more improvement (around **11%**). Finally, using GPT4 instead of GPT3.5 seems to be just as effective in improving the result (also around **11%** difference between those two). While this work focused on the Qald10 dataset, our system should also be able to improve the accuracy of any other kind of QA tasks, as long as the relevant entities mentioned in the question are present in Wikidata.

While the results themselves varies a bit between the different datasets, the same kind of conclusions can be reached for all of them. It does appear that including information from a knowledge graph (Wikidata) does lead to improved results in question and answer tasks. This is the case even when querying Wikidata directly (although the f1 score is a bit lower than when choosing the entity exclusively from entities present in the question's context). This implies that our system should improve the ability of the LLM to answer arbitrary questions, as long as those questions involves an entity present on Wikidata.

5.1. Future works

Both RAG and the Knowledge Graph Retrieval approaches have different strengths and weaknesses, combining them is a promising direction to improve performance. For example, using RAG to pair a Wikidata entity with chunks of text from it's corresponding Wikipedia article seems like a natural direction for improvements. This would improve the context fed to the model and allow it to answer questions even when information is missing from the KG. In other scenarios where entity names are not explicitly present in questions, RAG could assist in the difficult task of entity linking.

6. Contributions

Each member of the team contributed equally with regards to the project proposal, the midway report, the final presentation and the final report. Marc-André worked on the design, implementation and experiments for the KG part of the project, while Oscar and Michell did the same for the RAG part.

References

A. Perevalov, X. Yan, L. K. L. J. A. B. and Usbeck, R. Knowledge graph question answering leaderboard: A community resource to prevent a replication crisis. *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, (Preprint):2998—-3007, 2022.

D. Diefenbach, K. S. and Maret, P. Wdaqua-core0: A question answering component for the research community. *Semantic Web Challenges - 4th SemWebEval Challenge at ESWC 2017*, 769:84–89, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. The faiss library. 2024.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., and Wang, H. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

K. Shivashankar, K. B. and Steinmetz, N. From graph to graph: Amr to sparql. *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022)*, 2022.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Li, H., Su, Y., Cai, D., Wang, Y., and Liu, L. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*, 2022.

M.A.B. Santana, F. Ricca, B. C. and Barbara, V. Sparql-qa enters the qald challenge. *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022)*, 2022.

Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

N. Baramiia, A. Rogulina, S. P. V. K. and Razzhigaev, A. Ranking approach to monolingual question answering over knowledge graph. *Proceedings of the 7th Natural*

Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022), 2022.

Nussbaum, Z., Morris, J. X., Duderstadt, B., and Mulyar, A. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., and Wu, X. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*, 2023.

Perevalov, A., Diefenbach, D., Usbeck, R., and Both, A. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pp. 229–234. IEEE, 2022.

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., and Lewis, M. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Saxena, A., Tripathi, A., and Talukdar, P. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4498–4507, 2020.

Sun, L., Tao, Z., Li, Y., and Arakawa, H. Oda: Observation-driven agent for integrating llms and knowledge graphs. *arXiv preprint arXiv:2404.07677*, 2024.

Sun, Y., Shi, Q., Qi, L., and Zhang, Y. Jointlk: Joint reasoning with language models and knowledge graphs for commonsense question answering. *arXiv preprint arXiv:2112.02732*, 2021.

Talmor, A. and Berant, J. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*, 2018.

Usbeck, R., Yan, X., Perevalov, A., Jiang, L., Schulz, J., Kraft, A., Möller, C., Huang, J., Reineke, J., Ngonga Ngomo, A.-C., et al. Qald-10–the 10th challenge on question answering over linked data. *Semantic Web*, (Preprint):1–15, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

Wen, Y., Wang, Z., and Sun, J. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*, 2023.

Xu, K., Lai, Y., Feng, Y., and Wang, Z. Enhancing key-value memory neural networks for knowledge based 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)*, pp. 2937–2947, 2019.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yasunaga, M., Ren, H., Bosselut, A., Liang, P., and Leskovec, J. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*, 2021.

Yasunaga, M., Bosselut, A., Ren, H., Zhang, X., Manning, C. D., Liang, P. S., and Leskovec, J. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323, 2022.

Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., Cui, Y., Zhou, Z., Gong, C., Shen, Y., et al. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. arxiv 2023. *arXiv preprint arXiv:2303.10420*.

Zhang, X., Bosselut, A., Yasunaga, M., Ren, H., Liang, P., Manning, C. D., and Leskovec, J. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*, 2022.

Zheng, W., Yu, J. X., Zou, L., and Cheng, H. Question answering over knowledge graphs: question understanding via template decomposition. *Proceedings of the VLDB Endowment*, 11(11):1373–1386, 2018.