WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Master Thesis

Is GPT fit for KGQA?

Gerhard Georg Klager

Date of Birth: 21.10.1996
Student ID: 11801896

Subject Area: Information Business

Program Code: 066/960

Supervisor: Univ.-Prof. Dr. Axel Polleres

Date of Submission: March 12, 2024

Department of Information Systems & Operations Management, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR WIRTSCHAFTSINFORMATIK
UND OPERATIONS MANAGEMENT DEPARTMENT
OF INFORMATION SYSTEMS AND OPERATIONS
MANAGEMENT

Contents

1 Introduction **9**
- 1.1 Research Question . . . 10
- 1.2 Research Method . . . 10
- 1.3 Preliminary Results . . . 10

2 Preliminaries **11**
- 2.1 Semantic Web . . . 11
- 2.2 Knowledge Graphs . . . 11
  - 2.2.1 Wikidata . . . 11
  - 2.2.2 DBpedia . . . 13
- 2.3 Knowledge Graph Question Answering . . . 15
  - 2.3.1 Wikidata . . . 15
  - 2.3.2 DBpedia . . . 15
- 2.4 Artificial Intelligence . . . 16
  - 2.4.1 Symbolic vs. Sub-Symbolic AI . . . 16
  - 2.4.2 Neural Networks . . . 16
  - 2.4.3 Transformer Models . . . 17
- 2.5 Large Language Models . . . 17
  - 2.5.1 GPT-3 . . . 17
  - 2.5.2 GPT-3.5 . . . 18
  - 2.5.3 GPT-4 . . . 18

3 Related Work **18**
- 3.1 KGQA-Systems . . . 19
  - 3.1.1 Components of KGQA-Systems . . . 19
  - 3.1.2 Established Systems and Methods . . . 20
  - 3.1.3 Additional Research . . . 24
- 3.2 Benchmarking . . . 24
  - 3.2.1 QALD . . . 25
  - 3.2.2 SimpleQuestions . . . 25
  - 3.2.3 LC-QuAD 2.0 . . . 26
  - 3.2.4 WDAquaCore0Questions . . . 27
  - 3.2.5 StudentQuestions . . . 27
  - 3.2.6 WebQuestions . . . 27
  - 3.2.7 Additional Research . . . 28

4 Methodology **29**
- 4.1 Sample Benchmarks . . . 29
  - 4.1.1 Sample Questions . . . 29

4.1.2 Benchmark Analysis 37
4.2 Implementation of the Experiment 40
4.3 GPT-Based Components of KGQA 43
4.3.1 Question Analysis 43
4.3.2 Phrase Mapping 51
4.3.3 Disambiguation 54
4.3.4 Query Construction 55
4.3.5 Querying Distributed Knowledge 58

5 Results **58**
5.1 Question Analysis 58
5.1.1 Phrase Variation 59
5.1.2 Question Variation 60
5.1.3 Question Pseudonymization 62
5.1.4 Type Detection 63
5.1.5 POS Tagging 65
5.1.6 Named Entity Recognition 67
5.1.7 Entity and Relationship Detection 69
5.1.8 Semantic Parsing 70
5.2 Phrase Mapping 72
5.2.1 Direct Phrase Mapping 72
5.2.2 Global Phrase Mapping 74
5.3 Query Construction 78
5.3.1 Direct Construction 78
5.3.2 URI-Fed Construction 79
5.3.3 Template-Based Construction 81
5.3.4 URI-Fed Template-Based Construction 82
5.4 Technical Performance 83

6 Limitations **85**

7 Conclusion and Further Research **86**

A Is GPT fit for KGQA? â€“ Preliminary Results **95**

List of Figures

1 Example of the Wikidata data model for “United States of America”. 12
2 Example of the DBpedia data model for “President of the United States”. 14

List of Tables

1 Sample questions of the StudentQuestions benchmark. 30
2 Sample questions of the QALD-9 benchmark. 31
3 Sample questions of the SimpleQuestions benchmark. 31
4 Sample questions of the LC-QuAD 2.0 benchmark. 32
5 Sample questions of the WDAquaCore0Questions benchmark. 33
6 Average TTR for each benchmark sample dataset using GPT-3.5. 62
7 Average TTR for each benchmark sample dataset using GPT-4. 62
8 Number of question types classified by CBench per benchmark sample dataset. 64
9 Number of question types classified by GPT-3.5 per benchmark sample dataset. 64
10 Number of question types classified by GPT-4 per benchmark sample dataset. 65
11 GPT-3.5's POS tagging results for the question "What is the foundational document of the Soviet Union?". 65
12 GPT-3.5's POS tagging results for the question "Which is the Wikimedia category for the category of associated people of Oslo?". 66
13 GPT-3.5's POS tagging results for the question "washington square is a story by which American writer". 66
14 Resource identifiers generated by GPT-3.5 for the question "What is the foundational document of the Soviet Union?". . . 74
15 Resource identifiers and their label for the question "What is the foundational document of the Soviet Union?" generated by GPT-3.5. 75
16 Resource identifiers and their label for the question "who is the record label and genre of The_Velvet_Underground?" generated by GPT-3.5. 75
17 Resource identifiers generated by GPT-4 for the question "What is the foundational document of the Soviet Union?". 77
18 GPT-3.5's results of direct query construction by category. . . 78
19 GPT-4's results of direct query construction by category. . . . 79
20 GPT-3.5's results of URI fed query construction by category. . 80
21 GPT-4's results of URI fed query construction by category. . . 80
22 GPT-3.5's results of template-based query construction by category. 81
23 GPT-4's results of template-based query construction by category. 82

24 GPT-3.5's results of URI-fed template-based query construction by category. 83
25 GPT-4's results of URI-fed template-based query construction by category. 83
26 Average execution time per question for each task in the KGQA-process (in seconds). 84

Abstract

While research covering large language models such as OpenAI's GPT-models is on a sharp rise, their usage in Knowledge Graph Question Answering has barely been addressed. In this study, we therefore assess both GPT-3.5 and GPT-4's performance in some of the most commonly found tasks in KGQA. We find that both models show promising capabilities in classic NLP-tasks, and that GPT-4 is even capable of generating functioning SPARQL-queries based on a given natural language question. However, both models struggle with issues related to their reliability, in particular following given instructions and their API's stability. Additionally, we analyze sample sets of popular KGQA-benchmarks and find that no particular question type provides a harder challenge for the LLMs. Instead, we uncover problems lying within these benchmarks.

1 Introduction

The purpose of a Knowledge Graph Question Answering (KGQA)-system is to allow end-users to retrieve information stored in a KG by means of natural language questions, without being familiar with the KG's structure or the query language used to access said KG.

In order to achieve this goal, often some kind of translation of natural language questions into a query is taking place [16]. Many different KGQA approaches exist, ranging from template-based approaches [15] to approaches based on unsupervised message passing [60] or approaches using methods of machine learning [33]. The capabilities of KGQA-systems range from answering simple questions [13] to complex questions [67] as well as engage in single- and multi-turn (or conversational) question answering [69]. Additionally, some of these approaches even try to enable QA independent of a fixed KG or language [16].

To train and evaluate these models, numerous benchmarks have been created, enabling a direct comparison between existing and new QA-systems [43].

At the same time, with the recent success of OpenAI's ChatGPT[36] and its many competitors [23], we see many applications of such large language models (LLMs), not only restricted to question answering alone, but also in producing more or less useful code in programming and query languages.

With these rapid developments, uncertainty regarding the reliability and performance of these models [22, 29], as well as their future as a whole [3] is as relevant as their potential fields of application.

Facing these developments, we may ask ourselves both (a) if such LLMs can act as serious contenders to bespoke KGQA systems, and (b) whether LLMs could be used as a supportive technology for query formulation in the context of KGQA. However, literature covering this subject is still scarce and end-to-end QA-systems using LLMs such as ChatGPT in a synergistic combination with KGQA have not yet been proposed in abundance.

The aim of this thesis is therefore to fill this gap by exploring the possibilities of using LLMs such as ChatGPT in the task of KGQA and to challenge the status quo of existing benchmarks aimed at training and evaluating KGQA-systems.

1.1 Research Question

To fill the gap in the existing literature, wrt. LLMs in the context of KGQA this study aims at answering the following research questions:

1. How could LLMs be used to execute the KGQA-task.
2. Which components used in KGQA-systems could be enhanced using LLMs?
3. What types of questions are found in existing benchmarks for KGQA approaches, and in how far can these be used in benchmarking LLM-based QA-approaches?

1.2 Research Method

In order to answer these research question, we conducted a comprehensive literature review covering established KGQA-systems and their components, popular benchmarks used to evaluate these systems, as well as related work wrt. LLMs.

Following that, we conducted an experiment consisting of selecting a set of sample questions from some of the most popular benchmarks and evaluating these questions before establishing methods to use LLMs to execute some of the most commonly occurring tasks found in KGQA-systems.

Next, we use these methods to test the LLMs on the sample questions and analyze the results as well as the models' overall performances.

Last, we summarize the results and offer insights to further areas of research fit to extend the current state of the art of the usage of LLMs in the KGQA-process.

1.3 Preliminary Results

Note that preliminary results of this thesis have already been published in the Joint Proceedings of the Second International Workshop on Knowledge Graph Generation From Text and the First International BiKE Challenge co-located with 20th Extended Semantic Conference (ESWC 2023) [31].

In this preliminary study, we compared both GPT-3.5's and GPT-4's capability of answering questions of some of the most popular KGQA-benchmarks and a short dataset containing questions that, at the time, could not be answered correctly by the LLMs, as well as their capability of generating SPARQL-queries aimed at answering these questions over Wikidata.

The full paper can be found in the appendix A.

2 Preliminaries

This section is dedicated to provide an overview of the fundamental concepts that this study is based upon and therefore should be understood to comprehend the contents of this thesis.

2.1 Semantic Web

The Semantic Web or Web 3.0 has been an ongoing area of research for many years and in essence describes a machine-readable version of the World Wide Web [10]. Research on the topic of the Semantic Web covers aspects such as the transformation of existing documents into semantic web services [40], as well as many different languages and technologies related to it, such as the extensible mark-up language (XML), the Resource Description Framework (RDF) and the Web Ontology Language (OWL) [6]. Together with these languages and technologies, open knowledge bases or Knowledge Graphs such as Wikidata, storing highly interlinked data, received their spotlight in recent literature [21].

2.2 Knowledge Graphs

Knowledge Graphs (KGs) are knowledge bases using a graph-structured data model or ontology (i.e., a collection of interlinked descriptions of concepts, entities and their relations) to put this data into context by linking it and providing semantic metadata [42].

In this study we follow the definition used by Vakulenko et al. [60] in which a KG K defines a tuple $\langle E, G, P \rangle$ consisting of a set of entities E, as set of properties P and a set of labeled edges G. Both E and P are represented by uniform resource identifiers (URIs) connecting edges and properties ($\langle e_i, p, e_j \rangle \in G$, with $e \in E$ and $p \in P$).

2.2.1 Wikidata

Assembled from repositories in various fields, Wikidata is a community-maintained knowledge graph, adhering to the FAIR principles (findability, accessibility, interoperability and reusability) [62].

In Wikidata, any object or thing that can be considered, discussed, or observed is represented by so-called entities [47], which match to the Resource

Description Framework (RDF) schema [41]. Each entity (or statement) in Wikidata therefore represent a triple containing a subject called item, a predicate called property, and an object called value. These items and properties are identified using a Uniform Resource Identifier (URI) making them distinguishable from all other items and properties. A value can either be another item or a value of a certain datatype, such as a number.

In Figure 1 we visualize an example of this data model based on the first question in the StudentQuestions benchmark dataset used in this study "Who is the current president of the united states?".

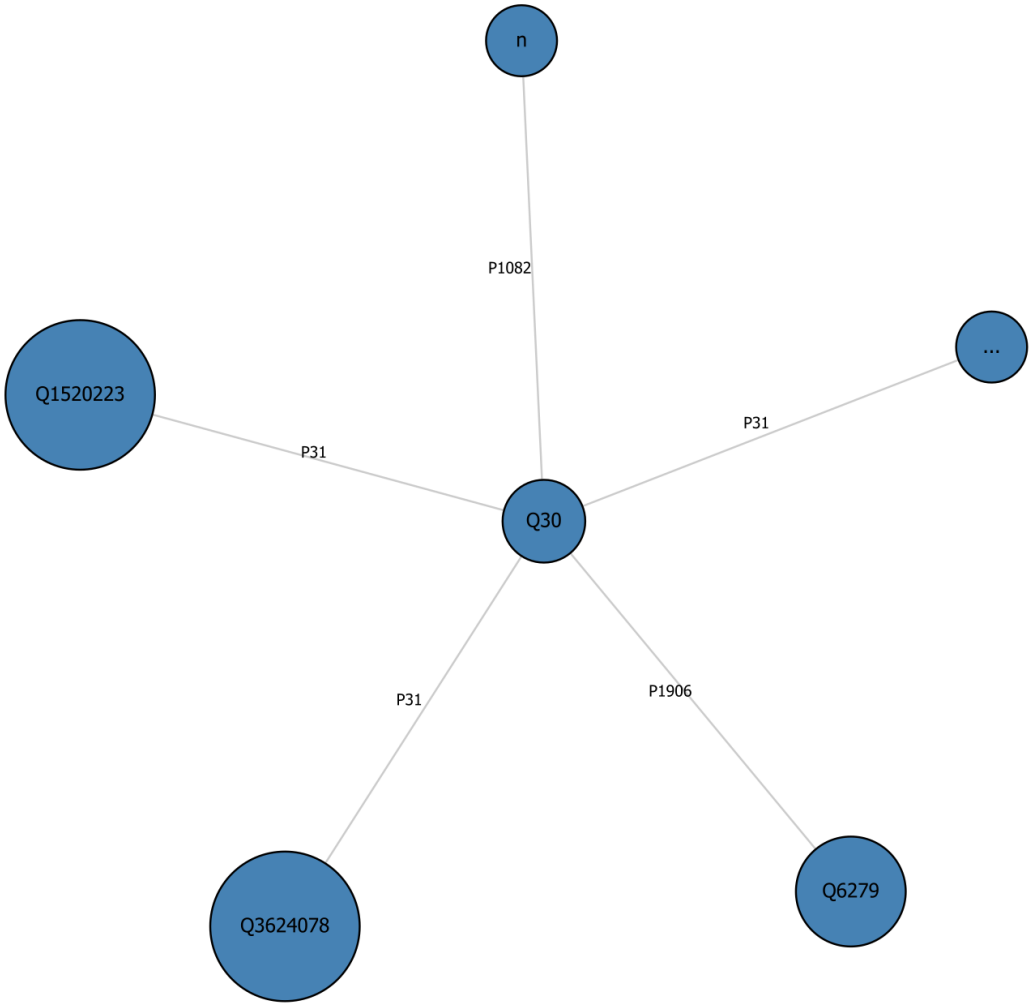

Figure 1: Example of the Wikidata data model for "United States of America".

Here, items and values are represented as vertices while the properties linking such items and values are represented as undirected edges between them.

The item "United States of America" (*Q30*) is connected to certain values over various properties. Over the property "is instance" (*P31*) the United States is not only an instance of the items "sovereign state" (*Q3624078*) and "constitutional republic" (*Q1520223*) but of many other items as well (...). On the other hand, the item "United States of America" is related to "Joe Biden" (*Q6279*) over the property "head of state" (*P1906*). Hence, Joe Biden is the head of state (i.e., president) of the United States of America. The property "population" (*P1082*) connects the item "United States of America" to the non-item value "3,929,214" (*n*). This number presents a "numerical" and has, in contrast to items, no URI to distinguish it.

2.2.2 DBpedia

DBpedia is another knowledge graph representing a crowdsourced community effort [1]. However, DBpedia consists of structured information extracted from Wikipedia.

Similarly to Wikidata, data on DBpedia can be represented and queried using RDF model. All entities in DBpedia are considered resources while properties can be categorized in ontology properties, which are part of the DBpedia ontology and non-ontological properties, which represent information found on Wikipedia and are used to represent factual data.

One of the main differences between DBpedia and Wikidata lies within their URIs. While Wikipedia's URIs consist of a letter followed by a number (*Q* for items and *P* for properties), DBpedia's URIs take the form of a URL, such ones commonly used for websites of various kinds. Since these URIs are not limited to numeric identifiers, they often contain information directly linked to the entity they represent. A similar example to the one shown for Wikidata in figure 1 is shown in figure 2.

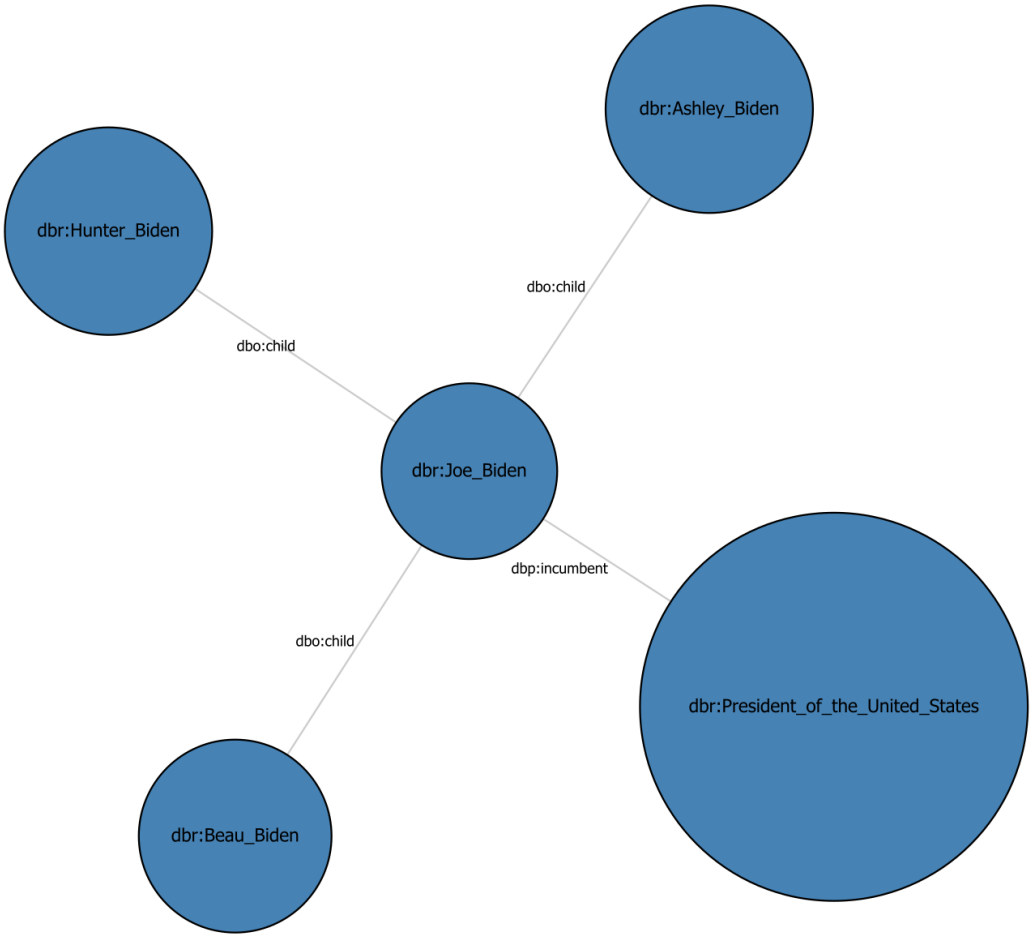

Figure 2: Example of the DBpedia data model for "President of the United States".

Here, the entity "Joe Biden" (*dbr:Joe_Biden*) is linked to the entity "President of the United States" (*dbr:President_of_the_United_States*) via the property "is incumbent of" (*dbp:incumbent*). The resources "Beau Biden" (*dbr:Beau_Biden*), "Hunter Biden" (*dbr:Hunter_Biden*), and "Ashley Biden" (*dbr:Ashley_Biden*) on the other hand are linked to "Joe Biden" over the property "is child of" (*dbp:child*).

Although simple, this example showcases DBpedia's intuitive naming convention wrt. its URIs.

2.3 Knowledge Graph Question Answering

"Knowledge Graph Question Answering" (KGQA) refers to the task of using the information stored in a KG to answer questions in natural language.

To query data from either Wikipedia or DBpedia the SPARQL-query language can be used. SPARQL is a query-language used to express queries for data stored (or viewed) as RDF [48].

2.3.1 Wikidata

For Wikidata, a query answering the question "Who is the current president of the United States?" could be constructing as shown in Listing 1.

```
SELECT ?P
WHERE {
    wd:Q1467287 wdt:P1308 ?P
}
```

Listing 1: Example query for Wikidata.

In this query, P represents the resource we want to identify, holding the value of property *P1308* (officeholder) for the item *Q1467287* (President-elect of the United States).

Note that items should be prefixed with wd: while properties should be prefixed with wdt: when querying Wikidata using SPARQL. However, only fixed values should be prefixed, hence the item we are searching for (P) does not receive a prefix.

2.3.2 DBpedia

Similarly to the Wikidata example, a possible answer to the question "Who is the current president of the united states?" could be retrieved using the query shown in listing 2.

```
SELECT DISTINCT ?P
WHERE {
    dbr:President_of_the_United_States dbp:incumbent ?P.
}
```

Listing 2: Example query for DBpedia.

Again, we want to determine the resource or in this case the president (*?P*) having the property "is incumbent of" (*dbp:incumbent)* for the resource "President of the United States" ("dbr:President_of_the_United_States").

2.4 Artificial Intelligence

Aimed at enabling computers to function in a way similar to the human brain and to be able to do the same things human minds can do [7], artificial intelligence (AI) covers a variety of fields, concepts and models.

2.4.1 Symbolic vs. Sub-Symbolic AI

AI is usually categorized in two different branches, namely symbolic and sub-symbolic AI [26].

Symbolic AI represents methods that produce logical conclusions and is also known as rule-based or knowledge-based AI. It relies on symbols (e.g., words, mathematical expressions) representing knowledge and rules set through human intervention to manipulate these symbols (e.g., encoded expert knowledge). Due to this rule-based nature, symbolic methods can be ineffective when used with noisy or large datasets.

Sub-symbolic, connectionist or neural network-based AI on the other hand aim at mimicking the human brain via a network of interconnected neurons. Relying on statistical learning methods and genetic algorithms, sub-symbolic methods are capable of learning and adapting based on new data. For this reason, sub-symbolic methods outshine its symbolic counterparts in dealing with large and noisy datasets where patterns and relationships cannot easily be defined via rules, such as image recognition or natural language processing.

Both symbolic and sub-symbolic methods find their usage ind knowledge graph tasks where symbolic methods can be used for tasks such as schema representation and triple classification while sub-symbolic methods are found to be used in predicting missing links in a knowledge graph or identifying the same entities across different knowledge graphs.

2.4.2 Neural Networks

Belonging to the branch of sub-symbolic AI and being inspired by the human brain, neural networks (NNs) describe computational models are made up of layers of interconnected nodes (artificial neurons) [32]. To process data, a so-called input layer matches each element in the input data (e.g., a number, a word in a text, a pixel in an image) to a specific neuron. Between this input layer and the output layer, a series of hidden layers is placed in which each neuron in each layer is connected by a weighted connection to each neuron in the previous, as well as the next layer. Via a so-called activation function, each neuron determines its output based on the sum of inputs it receives. Based on their shape (e.g., sigmoid, semi linear, rectified linear unit,

etc.), these activation functions allow for non-linearity in the model. In a step called forward propagation, the input data is being processed by being passed through the input layer and the hidden layers before reaching the output layer. The model is then being trained by comparing the output to a target output. Based on the difference computed by the loss function, between the real output and the target output (the so-called loss), the weights within the network are being updated in a way that minimizes this loss (backward propagation). After updating the weights, the input data is fed through the model and the loss is computed once again. This process will be repeated until the model is deemed accurate enough. To use the model on new data, the last set of weights will be used, and the input will be sent through the model only once.

2.4.3 Transformer Models

Transforming a sequence of inputs into a sequence of outputs (also called sequence transduction) is the core of many machine learning tasks, such as speech recognition, machine translation and text-to-speech [25].

To solve these tasks, so-called transformer models (or transformers) have been developed. A transformer represents a type of neural network architecture based entirely on attention mechanisms [61]. Resulting not only in higher quality results than traditional neural sequence transduction models, but also requiring less time to train and being more parallelizable, transformers make use of the possibility to draw global dependencies between input and output, yet not relying on recurrence and convolutions.

2.5 Large Language Models

Large Language Models (LLMs) are AI models capable of processing and generating human-like language [11]. LLMs are usually trained on large amounts of text data and make use of transformer models in order to learn patterns and relationships between words and phrases in a text.

In this section, we provide an overview of how LLMs function and the LLMs used in this study, as well as other literature surrounding them relevant to this study.

2.5.1 GPT-3

With over 175 billion parameters, GPT-3 was one of the largest autoregressive language models of its time [9]. Able to generate human-like text and

completing a variety of tasks including translation, question-answering and performing on-the-fly reasoning or domain adaptation, the model performs well on many NLP datasets. The model was pre-trained on a large corpus of text data and, using a self-supervised learning approach, it learns to predict the next word in sequence based on the previous words.

2.5.2 GPT-3.5

OpenAI's GPT-3.5 model series is an upgrade to the GPT-3 based models and is based on the code-davinci-002 model released for code generation tasks [68]. The GPT-3.5 series models were trained by the usage of supervised fine-tuning and the Illustrating Reinforcement Learning from Human Feedback (RLHF) training strategy.

Note that an initial comparison of GPT-3 and GPT-3.5 for KGQA has already been done in our previous work [31].

2.5.3 GPT-4

Released in March 2023, OpenAI's GPT-4 provided a significantly further developed alternative to the GPT-3.5 based models. The model is able to outperform humans on various tasks and was able to be among the best 10% of test takers for a simulated bar exam. Besides improving GPT-3.5's performance, wrt. general knowledge and its reasoning capabilities the developers also directed their resources at improving the model wrt. its adherence to desired behavior and various safety concerns, such as the generation of harmful content.

Additionally, GPT-4 can not only generate code but test it as well and therefore exceeding the capabilities of classic transformer models.

While not implemented at the time of writing, GPT-4 has furthermore been designed to accept not only textual input but also images in the future [2].

3 Related Work

With the growing attention given to KGQA in recent years, one can also observe a large growth of literature covering KGQA-systems and in terms of different methods and benchmarks to evaluate these systems. This section is dedicated to providing an overview of this literature and laying out the foundation for our planned research.

3.1 KGQA-Systems

To answer natural language questions using linked data stored on KGs a variety of different approaches and KGQA-systems have been created and covered in academic literature. This section aims at providing an overview of some of the existing KGQA-systems, their components, as well as other studies evaluating them and discussing some of their characteristics.

3.1.1 Components of KGQA-Systems

To solve the multitude of challenges related to KGQA, KGQA-systems usually combine various techniques ranging from natural language processing, information retrieval and machine learning to Semantic Web, Diefenbach et al. conducted a survey covering numerous KGQA-systems aimed at describing the techniques commonly used in the KGQA-process [17].

The KGQA-systems used in their study were evaluated over the popular QALD benchmark series, as well as the WebQuestions and the SimpleQuestions benchmarks. The authors grouped the various techniques used in these systems into the five categories "Question Analysis", "Phrase Mapping", "Disambiguation", "Query Construction" and "Querying Distributed Knowledge".

Question Analysis
During the "Question Analysis" phase, syntactic features of the natural language question are used to extract information about the question, such as determining the question type, identifying named entities, and identifying entities, relations and their dependencies. Common techniques used in this category are techniques to recognize named entities, such as entity recognition and entity linking approaches, and N-gram strategies, segmenting the question using part-of-speech (POS) tagging approaches using handmade or learning rules and parsing techniques based either on the phrase structure grammars or dependency grammars to identify dependencies.

Phrase Mapping
The task of "Phrase Mapping" deals with finding resources that have a high likelihood of corresponding to a given phrase or word in the natural language question. Techniques of "Phrase Mapping" use Knowledge Base labels, dealing with misspelling using string similarities such as measurements of the distance between a phrase and the labels of resources in the KG, and measures to deal with semantic similarities, such as using databases with lexicalizations and redirects.

Disambiguation
The "Disambiguation" deals with ambiguous segmentations and dependencies resulting from the question analysis step and with multiple possible resources for a single phrase returned by the "Phrase Mapping" step. Techniques in this step range from local disambiguation and graph search techniques using the structure of the KG to hidden Markov models, integer linear programs and even the incorporation of user feedback.

Query Construction
"Query Construction" is the task of constructing the corresponding query to a given natural language question. During this task the problem of the "semantic gap" arises which refers to the fact that the KG might encode an information differently than one could deduce from the question making it in theory impossible to deduce a SPARQL queries using only the natural language question. The techniques used in this step range from filling in SPARQL templates to using information gained from the previous steps such as the results of the "Phrase Mapping" phase, semantic parsing approaches to machine learning approaches and even approaches that do not use SPARQL.

Querying Distributed Knowledge
Last, "Querying Distributed Knowledge" covers the aspect of using multiple KGs to answer a single question instead of one. The approaches related to this setting either assume the KGs to be either disjoint graphs or interlinked, meaning that the resources on the different KGs representing the same entity are linked.

3.1.2 Established Systems and Methods

Diefenbach et al. proposed a QA-system capable of querying multiple KGs independent of the natural language used [16]. Their approach has been evaluated on five well-known KGs and five different languages using three different benchmarks. This proposed QA-system first performs entity recognition in terms of searching corresponding international resource identifiers (URIs) whose lexicalization forms an N-gram (i.e., consecutive elements in a text) in the asked natural language question. After removing stop words from the set of URIs, queries that could represent possible interpretations of the question are constructed before being ranked based on multiple aspects, such as the number of words matching the words in the original question. Next, a logistic regression based on labeled SPARQL-queries will be trained to compute a confidence score for each query. Last, the highest ranked query above

a certain threshold will be used to answer the question. If no query with confidence above this threshold is found, the whole question will be deemed unanswerable. During their study, the authors discovered performance differences in their approach wrt. different (natural) languages used, and link these differences to the quality of the available data for each language.

Vakulenko et al. took a different approach based on the usage of unsupervised message passing [60]. Their approach, named QAmp, consists of two phases: in the first phase called question interpretation, the relevant sets of entities and predicates necessary for answering the input question are again being identified, and their confidence scores are being computed. In the second phase, the so-called answer inference phase, these confidence scores a propagated and aggregated over the underlying KG's structure, providing a confidence distribution over a set of possible answers which is then be used to locate the corresponding answer entities, rather than translating the query to SPARQL.

Yani et al. propose a method to detect entities and their position on triples that have been mentioned in a complex question [67]. Their approach is capable of not only detecting the entity name, but also of determining in which triple the entity is located and if the given entity is a head or tail of the triple. In their approach, Yani et al. split a natural language into tokens and detect the position of triples in the question before finding the head and tail positions in the underlying KG. This process is split into two phases.

In the position-based pattern set construction, a question-answer pair is used as an input to obtain the triple set in the KG expressing the answer to the question. Afterward, a position-based pattern set will be constructed for the question.

During the classification of the input question, words are being grouped into synonyms following a Wordnet-based approach. These synonyms are then being explored to find verbs and nouns before generating synonym questions. Last, using a multi-class classifier with an underlying transformer model the position-based patterns of the question are being predicted.

Another QA-system proposed by Liang et al. is based on the idea of splitting the process of translating natural language questions into SPARQL-queries into five sub-tasks [33]. First, a random forest model is trained to identify a question's type. Next, various entity recognition and property mapping tools are used to map the question's phrases before all possible triple patterns are created based on these mapped resources. Afterward, possible SPARQL are generated by combining these triple patterns into a

based on the question's type, before a Tree-LSTM based ranking model is used to select the most plausible SPARQL query representing the correct intention behind the natural language question. Possible SPARQL queries are then constructed by combining these triple patterns in the query generation step. In order to select the correct SPARQL query among a number of candidate queries for each question, a ranking model based on Tree-LSTM is used in the query ranking step. The ranking model takes into account both the syntactical structure of the question and the tree representation of the candidate queries to select the most plausible SPARQL query representing the correct intention for the respective question.

The pipeline framework for KGQA proposed by Cui et al. focuses on single-relation questions (questions that can be answered through a single fact in the KG) and is divided into three components [13].

The entity detection model is tasked with labeling the mentioned entities in the question. This is done by viewing the question as a sequence of tokens and assigning a label by using a combination of a bi gated recurrent units (BiGRU) network, as well as a conditional random field (CRF) network.

To generate candidate entities for each token, the entity linking model, based on Mohammed et al. [38], uses a pre-build inverted index mapping all N-grams of an entity to its name before extracting all N-grams from the entity mention and generating an entity candidate set via matches in the inverted index. Last, these entity candidates are then ranked via their Levenshtein distance.

Finally, an attention-based model consisting of a relation encoder, a question pattern encoder and a similarity measure is used to measure the semantic similarities between the question and its corresponding relation candidates.

Due to the linguistic differences between the Korean language and other languages, Jung and Kim proposed a method to translate Korean natural language queries into SPARQL queries, in order to extend the field of ontology-based QA wrt. the Korean language [30]. In their approach, the input natural language query is first being split into tokens before mapping each token to certain resources in the ontology. Afterward, multiple query graphs are generated from these mappings by arranging resources and identifying the relationships between them using a path search algorithm based on the domain ontology schema's structure. Finally, these query graphs are scored based on how accurately they reflect the user's intent, and the query graph with the highest ranking is being converted into a SPARQL-query.

Shin et al. noticed that QA systems suffer notably from the divergence of

the unstructured data composing natural language questions and the structured data composing a KG [54]. Existing approaches to deal with this issue use lexicons in order to cover differently represented data. Since these lexicons only consider representations for entity and relation mentions, the authors propose a new predicate constraint lexicon restricting subject and object types for a predicate. This so-called Predicate Constraints based Question Answering (PCQA) lexicon does not make use of any templates. Rather, the authors generated query graphs focusing on matching relations in order to cover diverse types of questions.

A more complex task in the field of KGQA is multi-hop KGQA, which requires reasoning over the multi-hop relational chain within the underlying KG in order to answer the natural language question [28]. Multi-hop questions in this context define natural language questions involving more than one predicate between their topic entity and their answers. The proposed Relational Chain based Embedded KGQA (Rce-KGQA) model makes simultaneous use of both the explicit relational chain observed in a natural language question, and the implicit relational chain within the underlying KG.

In the so-called Answer Filtering Module, filters a set of possible answers (entities) from the KG via the methods of graph embedding, semantic parsing, as well as answer scoring.

The Relational Chain Reasoning Module on the other hand is dedicated to improving the reasoning accuracy by consideration of the reasoning chain order and its relational type.

To counteract inefficiencies rooted in relying on question understanding and conventional graph-based algorithms, Wang et al. proposed a framework to construct queries via Knowledge Graph Embedding [63].

In their approach, the underlying KG is first being encoded into a low-dimensional embedding space using generalized local KGs.

This embedding representations are then used to compute the query structure and assemble vertices and edges into the query (i.e., generate the query).

(AQQUCN) is a QA system combining the usage of KGs and corpus evidence [53]. Instead of relying on one semantic interpretation of the query, the system aggregates signals from both KGs and large corpora to rank entities within the KG. For this, the question's ideal interpretation is modeled as a latent variable and its interpretations and candidate entities are being scored as pairs.

3.1.3 Additional Research

Conversational Question Answering (CQA) is a process in which the CQA-system must be able to understand questions asked in the context of the previously asked questions [69]. Zaib et al. conducted a survey in their effort to provide a comprehensive review of the state-of-the-art research trends in the field of CQA.

Covering over 80 conference and journal papers, the authors acknowledged a trend from single-turn to multi-turn QA and categorize CQA-systems based on the data domain, the types of questions answered, the data sources and the types of systems built for the question at had.

The authors deem CQA-systems to be potentially used in different types of QA such as open-domain or closed-domain QA (e.g., in the medical field), as well as in various commercial areas.

While simple questions can be answered with a subject-predicate-object triple, complex questions require more information and the usage of more advanced query operations to be answered. In their study, Gomes et al. aim at providing an overview of the methods used to answer such complex questions over knowledge bases [24].

The authors divide complex questions into multi-hop and constraint questions. While multi-hop questions require the combination of information from multiple facts in the KG to be answered, additional constraints beyond the original questions must be considered by the QA-system to answer constraint questions.

Additionally, they find that most approaches are either based on neural networks or semantic parsing. Additional, so-called neural network-based semantic parsing, which presents a combination of the two methods, established itself as the state of the art.

3.2 Benchmarking

In order to evaluate KGQA-systems such as the ones mentioned in section 3.1.2 numerous benchmarks have been created. This section aims to provide an overview of and describe the benchmarks used in this study, as well as of some of the most popular and important benchmarks used in research relating to KGQA. We furthermore discuss the properties of each of the used benchmarks and point out noteworthy observations.

Additionally, we cover existing research focusing on comparing, evaluating and constructing benchmark datasets for KGQA.

3.2.1 QALD

The probably most widely used family of datasets for Question Answering is represented by the Question Answering over Linked Data (QALD) campaign. This series of challenges aims at providing multilingual benchmarks for all QA-systems designed for using natural language requests of a user to retrieve information stored as structured data, such as the RDF data format. Additionally, the challenge aims at comparing current state-of-the-art QA-systems wrt. their individual strengths and shortcomings. In order to participate in the current QALD challenge, users can simply run their QA-system using the current challenge's dataset before storing their results in an XML-file and upload it to the challenge's website [34].

At the time of writing, the QALD challenge is currently taking place in its 10^{th} iteration, which laid its focus on Wikidata. The reason for this lies within the ongoing shift from the popular Knowledge Graphs Freebase and DBpedia to Wikidata occurring in research. This shift is rooted in Freebase being defunct, since it was shut down by Google, and DBpedia lacking the structural validity of Wikidata. The new dataset therefore adapted the existing corpus, which was based on DBpedia by increasing its size and adjusting it to Wikidata's ranking mechanism of properties [50].

While the 10^{th} iteration of the QALD challenge with its focus Question Answering over Wikidata might be able to provide valuable insights for our study when used, we decided to use the 9^{th} iteration of the challenge instead [59]. The reason for this is twofold. First, we did not experience a lack of benchmarks to assess the effectiveness of the GPT models on Wikidata. Second, since the QALD-9 challenge is already completed at the time of writing it not only provides a complete set of questions and their corresponding sample solutions, but potential faults within the dataset might have already been discovered at this point.

The QALD-9 dataset contains not only its questions in multiple languages but also provides their corresponding keywords, answer types, sample solution queries, as well as other information related to them.

3.2.2 SimpleQuestions

SimpleQuestions is a dataset containing 100k questions aimed at training and evaluating QA-systems wrt. solving the simple question answering problem, which consists of answering questions that can be rephrased as (single triple) queries that ask for all objects linked to a question's given subject by the subject's given relationship. In this context, simple QA is a term used to refer to the simplicity of the reasoning process necessary to answer questions

[8].

While SimpleQuestions was originally designed to be used over Freebase, Diefenbach et al. have adapted the original SimpleQuestions dataset in order for it to be usable over Wikidata [19].

While the benchmark is close to being solved and some of its questions clearly have more than one correct interpretation, it remains one of the most popular datasets used [45].

Besides a variety of questions, the dataset also includes the corresponding triple consisting of a subject, a predicate and an object for each question, where the object represents the answer to the given question.

3.2.3 LC-QuAD 2.0

As an alternative to simple questions, the Large-Scale Complex Question Answering Dataset 2.0 [20] (LC-QuAD 2.0) is an extension to the original LC-QuAD dataset [57] containing 30k complex questions as well as their corresponding paraphrased versions and SPARQL queries. The dataset is both compatible with Wikidata and DBpedia (2018) containing 21,258 unique entities and 1,310 unique relations. The dataset was created by generating a number of SPARQL queries before verbalizing them into natural language questions using the Amazon Mechanical Turk. Afterward, these questions have been paraphrased to create additional natural language questions. The questions in the LC-QuAD 2.0 benchmark dataset are categorized into ten different question types. "Single fact" questions can be answered using queries over a single fact (subject-predicate-object) which could return either a subject or an object. "Single fact with type" questions include constraints in a single triple query. "Multi-fact" questions can be answered using queries over two connected facts in Wikidata. "Fact with qualifiers" questions use additional properties for facts stored in the KG to make more informative questions. "Two intention" questions represent questions posing two intentions, while "boolean" questions can be answered using only "true" and "false" statements. Additionally, "count" queries perform a count over how often the predicate is used with an entity or an object using the "COUNT" keyword in SPARQL. "Ranking" questions ask for entities with either a maximum or a minimum value of a given property. "String Operation" questions ask about entities at word level or at character level. Last, "temporal aspect" questions cover temporal properties in both the question space, and the answer space.

3.2.4 WDAquaCore0Questions

Another approach of creating a benchmark for KGQA has been used for the WDAquaCore0Questions dataset. This dataset consists of questions that have been asked by users of the demo of the WDAquaCore0 QA-system [18] and corresponding sample solution queries that have been collected using the feedback function [14]. While the original QA-system was able to be used for both DBpedia and Wikidata [18] most of the provided SPARQL-queries for questions in the benchmark are written to work on Wikidata.

3.2.5 StudentQuestions

In light of recent developments, and with social media being full of examples, there is — to the best of our knowledge — not yet a dedicated QA dataset originally tailored to LLMs and GPT specifically. In order to fill this gap, we asked students of the Digital Economy masters' program at the Vienna University of Economics and Business to generate a set of natural language questions aimed at asking ChatGPT to formulate queries GPT-3 would fail upon but suspected to be possible to answer with the information in publicly available KGs such as Wikidata. As a hint, we emphasized that we suspect LLMs to struggle with recent events' information beyond the training phase of the LLM, as well as complex questions that require non-obvious conceptual understanding and reasoning. The students' task was to further formulate the corresponding SPARQL queries and – in the light of recent advances of LLMs for code and query generation – attempt to write prompts that would lead ChatGPT to be able to create such queries.

This experiment resulted in a dataset containing 13 questions, as well as their corresponding sample solution SPARQL queries for Wikidata. Additionally, we added the question "Who is the current president of the united states?" to the dataset in order to provide a question following a simple structure, for which we assume that Large Language Models should be able to generate a functioning SPARQL query for.

3.2.6 WebQuestions

Berant et al. [5] created a new QA dataset named WebQuestions. The WebQuestions dataset acts as an extension to the FREE917 dataset (based on Freebase) aimed at evaluating QA-systems. The authors created this dataset due to the FREE917 dataset [29] requiring logical forms, making it inherently more difficult to scale it up due to the requirement of having expertise in annotating logical forms. Using the Google Suggest API, the authors obtained questions beginning with a wh-word (where, who, when,

etc.) and containing exactly one entity. For each question, five candidate queries have been created. After collecting 1M questions in this process, 100k randomly selected questions have been submitted to Amazon Mechanical Turk where workers answered questions, detecting duplicates and filtering out questions that could not be answered. The remaining dataset contained 5,810 questions.

3.2.7 Additional Research

Jiang and Usbeck analyzed 25 KGQA benchmark datasets, wrt. five different KGs namely DBpedia, EventKG, Freebase, Wikidata, and WikiMovies. Their study showed that many available KGQA datasets are unfit to train KGQA-systems due to their underlying assumptions or because these datasets are outdated and based on discontinued KGs, such as Freebase. Additionally, the authors share light on the difficulties and high costs related to the generation of new datasets for benchmarking KGQA-systems. Therefore, they propose an automated method to re-split existing datasets and enabling their generalization, as well as a method to analyze existing KGQA datasets with regard to their generalizability [27].

While many different benchmarks aimed at evaluating QA-systems for different KGs exist the question of which benchmark one should use can be a difficult one to answer. To answer this question Orogat, Liu, and El-Roby proposed CBench, a suite that enables users to analyze either the 17 benchmarks already included in CBench or any benchmark added by the user with regard to linguistic, syntactic, and structural properties of the datasets' questions [43].
The analysis of natural language questions covers for example the determination of the questions' types. These types consist of "wh"-questions which represent questions starting with a "wh"-pronoun (i.e., What, When, Where, Who, Whom, Which, and Whose), "how"-questions marking questions starting with the keyword "how", "yes/no" questions that can be answered with "yes" or "no", "request" which represent rephrased "what"-questions but are usually handled differently in QA-systems, and "topicalized" questions, topicalizing entity or prepositional phrases for the purpose of emphasis.

The analysis of structured queries on the other hand analyzes for example the frequency of the operators used in queries, the percentage of keyword occurrences in the queries of a given benchmark, as well as an analysis of the structural shape of the queries.

Additionally, CBench can be used in order to evaluate a KGQA-system over a given benchmark dataset.

Last, the authors provide an overview of different creation methods for benchmarks, ranging from manual creation based on heuristics to benchmarks created automatically from the KG in question.

In this study, we use CBench to analyze both the questions in our selected benchmark datasets, and their corresponding sample solution queries.

4 Methodology

In order to determine the GPT-models' fitness for KGQA we first determined individual components or tasks commonly occurring in KGQA. Next, we drew a random sample of each benchmark dataset before developing methods to let the previously defined tasks be executed by the GPT-models before analyzing the obtained results and comparing them to the benchmark datasets' sample solutions. Similarly to the previously done assessment of the GPT models' performances, wrt. the task of KGQA as whole [31] the R programming language [49] has been used to generate the results of the assessment of GPT models' performances, wrt. the different components found in common KGQA-systems. Last, we conclude in which areas the GPT models returned promising results, where the models showed to be strongly limited, and how these limitations could be overcome in further research.

4.1 Sample Benchmarks

To evaluate the LLMs' performance on the KGQA tasks, we analyzed the benchmarks mentioned in section 3.2 and randomly drew a set of sample questions for each one of them.

4.1.1 Sample Questions

To evaluate the performance of the tested GPT models, we selected a multitude of benchmark datasets out of the ones mentioned in section 3.2 and limited each one to a maximum 15 random questions. The datasets have been selected based on both their popularity, and their availability. Limiting the number of questions per benchmark dataset was done for the following reason: Since the assessment of the GPT models' performances, wrt. each task in a typical KGQA-process is a manual and qualitative task itself, the number of questions and therefore the number of results had to be chosen in a way that ensures that the resulting findings remain meaningful and generalizable while being small enough to obtain these findings during a reasonable time frame. Listing 3 shows the R-code used to sample the questions

where $jData$ is the original benchmark dataset, n is the number of questions wanted, and 47 is a random number selected by ChatGPT used as seed to keep the results reproducible.

```
#### Limiting to n random questions ----
n <- 15
set.seed(47)
jData <- jData[[2]]
jData <- jData[sample(1:length(jData), n)]
```

Listing 3: R-code used to sample benchmark questions.

The selected questions for each of the benchmark datasets used in this study are shown in tables 1 to 5.

Table 1: Sample questions of the StudentQuestions benchmark.

	Question
1	Who is the current president of the united states?
2	Who won the football worldcup 2022?
3	Give me all Austrian female actors that are aged over 50.
4	Give me all Austrian female actors aged over 50 years that are also dancers or singers.
5	When did the famous Brazilian football player Pelá
6	For which team does Lionel Messi play?
7	What is the most recent MineCraft Java Edition version?
8	How many people do live on earth?
9	What was the average temperature in Vienna in 2022?
10	Who is the fastest person in the world?
11	What is the oldest painting in the world?
12	Where does the handball world cup take place this year (2023)?
13	Who is CEO of Twitter?
14	Which team won the 'Serie A' championship last season?

Table 2: Sample questions of the QALD-9 benchmark.

	Question
1	Who is the daughter of Ingrid Bergman married to?
2	How often did Jane Fonda marry?
3	Give me a list of all lakes in Denmark.
4	Who are the developers of DBpedia?
5	Who composed the soundtrack for Cameron's Titanic?
6	Give me the grandchildren of Bruce Lee.
7	How many inhabitants does the largest city in Canada have?
8	Was the Cuban Missile Crisis earlier than the Bay of Pigs Invasion?
9	Who is the president of Eritrea?
10	In which country does the Ganges start?
11	What is the timezone in San Pedro de Atacama?
12	Who writes the Farmers' Almanac?
13	What is the birth name of Adele?
14	Show me all songs from Bruce Springsteen released between 1980 and 1990.
15	Give me all actors who were born in Paris after 1950.

Table 3: Sample questions of the SimpleQuestions benchmark.

	Question
1	Is the sex of anuradha sriram male or female?
2	where was bob stewart born?
3	what kind of music did bret michaels make
4	which country is antonio dixon from
5	What is anvar rajabov's gender
6	what book was written by rabindranath tagore?
7	what bridge is a cable-stayed bridge?
8	what is destro's gender?
9	what coast does was nominated for Classical Album of the Year
10	Who directed ghajini?
11	Where is william watt from?
12	what is oliver holzwarth's gender?
13	what is the producing company of the movie lamhe
14	what labels has ron white signed to
15	washington square is a story by which American writer?

Table 4: Sample questions of the LC-QuAD 2.0 benchmark.

	Question
1	What is the foundational document of the Soviet Union?
2	On which island is the HQ location of the Carlsberg Group?
3	What volcanic eruption occurred in the Dutch East Indies?
4	What is the architecture firm that is based in Saint Longinus?
5	who is the record label and genre of The_Velvet_Underground?
6	What is Loop ID for Simon Baron-Cohen?
7	What is the religious affiliation of the victim of the Battle of Stalingrad?
8	What time did Aarhus serve as an adminstrative body at Rostock?
9	What is the position of political office held by a member of Augustus' family?
10	What are the beliefs of the Chinese Communist Party's Chair, Hu Jintao?
11	Which is the {international sport governing body} for {authority} of {baseball}?
12	What are the plays of the organizer of the UMB World Three-cushion Championship?
13	Which is the Wikimedia category for the category of associated people of Oslo?
14	When did Battle of Quiberon Bay happen?
15	How many country citizenship are taken by Antonio José de Sucre Farell?

Table 5: Sample questions of the WDAquaCore0Questions benchmark.

	Question
1	which currency lyon
2	zell teilung
3	in which continent is slovenia
4	When was J. K. Rowling born?
5	who is barack obama
6	university of ottawa
7	who won the eurovision song contest 2016
8	part of a thunderstorm
9	Give me all the german writers.
10	lyon
11	tagesschau website
12	is barck obama's wife name michelle?
13	who leonardo dicaprio
14	food france
15	london

Additionally, a set of further preparations have been done in order to enable the assessment of the GPT model's fit for executing the various tasks within KGQA.

While most of the selected benchmark datasets included a sample solution SPARQL query for each question, two benchmarks required some efforts in this direction.

Since the StudentQuestion dataset has been created as an in-class experiment aimed at finding questions unanswerable by ChatGPT that can be answered via SPARQL and Wikidata, it already included sample solution queries. However, for some of the questions, these queries were either faulty or did not exist. We therefore manually corrected or added the queries in question.

The SimpleQuestions benchmark did not provide any sample solution queries at all but added the corresponding object, predicate, and subject triple from which we were able to construct each sample solution query for our subset. Again, we did this manually. Since the original benchmark dataset came with wrongly labeled predicate-identifiers, we corrected these keywords manually.

The QALD-9 benchmark dataset already provides the keywords for each question, while others do not. For this reason, we added the keywords for each question to each benchmark dataset. This was done manually for the

StudentQuestion, SimpleQuestions and the WDAquaCore0Question benchmark datasets, by selecting the relevant keywords in each question. The selected keywords therefore stem out of the authors' own subjective interpretation and therefore only provide a loose idea as to what for example by the GPT models' extracted keywords should be compared against. Since the LC-QuAD 2.0 benchmark already highlighted the keywords for most questions by wrapping them in curly brackets, we extracted these keywords using regular expressions. The question "What is the {architecture firm} that is {based in} {Saint Longinus} ?" for example. Its corresponding keywords would therefore be "architecture firm", "based in" and "Saint Longinus". Listing 4 shows the R-code for this step.

```
# Adding keywords
dfResWD$Keywords <- NA
for(q in 1:nrow(dfResWD)){
  cKeywords <- jData[[q]]$NNQT_question
  cKeywords <- unlist(str_extract_all(cKeywords, "\\{.*
     ?\\}"))
  cKeywords <- gsub("[{}]", "", cKeywords)
  cKeywords <- paste(cKeywords, sep = "", collapse = ", "
     )
  dfResWD$Keywords[q] <- cKeywords
}
dfResWD$Keywords[6] <- "Loop ID, Simon Baron-Cohen"
```

Listing 4: R-code used to extract the keywords for the questions of the LCQuAD 2.0 benchmark dataset.

Note that no keywords were highlighted using curly brackets for the question "What is Loop ID for Simon Baron-Cohen?". We therefore manually added "Loop ID" and "Simon Baron-Cohen" as keywords for this question.

Last, we ensured that all of our benchmark dataset samples include the corresponding URIs for keywords/named entities for each sample solution query to be able to pass these as input to the GPT models during the query construction tasks.

Since the naming conventions for DBpedia's SPARQL query prefixes was inconsistent in our subset of the QUALD-9 benchmark dataset, we did not construct these URIs in an automated manner, but rather did so manually ourselves. For the question "Who is the daughter of Ingrid Bergman married to?" an example of these URIs would be "dbr:Ingrid_Bergman, dbo:child, dbo:spouse" where "dbr:" represents a DBpedia-resource and "dbo:" represents a DBpedia-ontology. DBpedia-concepts are represented by the prefix

"dbc:".

For the SimpleQuestions benchmark dataset, we simply used the subject and predicate identifiers provided by the dataset. We excluded the object identifiers since these represented the solutions to the questions and might therefore distort the results of this study. Since Wikidata's URIs are of a numeric type, we cannot assume that the GPT models will know which term is represented by these URIs. We therefore retrieved the labels of each URI using Wikidata's query service in R and linked these to URIs provided by the dataset. Listing 5 shows the R-code extracting the URIs for the SimpleQuestions benchmark dataset.

```
#### Adding column with URIs
dfRes$URIs <- ""
for(i in 1:nrow(dfRes)){
  for(j in 2:3){
    cQuery <- paste0(
      "SELECT ?name WHERE { wd:",
      dfRes[i, j],
      "rdfs:label ?name. FILTER (LANG(?name) = \"en\").}"
    )
    cName <- query_wikidata(
      sparql_query = cQuery,
      format = "simple"
    )
    cName <- unlist(cName)
    dfRes$URIs[i] <- paste0(dfRes$URIs[i], dfRes[i, j], "
        = ", cName, ", ")
  }
  dfRes$URIs[i] <- substr(dfRes$URIs[i], 1, nchar(dfRes$
     URIs[i]) - 2)
}
```

Listing 5: R-code used to extract the URIs for the questions of the SimpleQuestions benchmark dataset.

Since the SimpleQuestions benchmark datasets includes a column for the questions' predicate-URIs, as well as one for their subject-URIs, this code extracts the URIs of the respective columns in the benchmark data frame and links them to their corresponding label retrieved over Wikidata's query service. Listing 6 shows the resulting query for retrieving the subject label for the question "what kind of music did bret michaels make".

```
SELECT ?name WHERE {
    wd:Q365042 rdfs:label ?name.
```

```
    FILTER (LANG(?name) = "en").
}
```

Listing 6: SPARQL-query for Wikidata to retrieve the subject label of the question "what kind of music did bret michaels make" in the SimpleQuestions benchmark.

Note that some of the sample identifiers have been labeled incorrectly (see section 4.1.2). Using this method, it was not possible to add labels to the wrongly labeled predicate-identifiers mentioned earlier. Despite the easy fix of replacing the letter R in the predicate identifiers with the letter Q before labeling these identifiers, we opted for not doing this in order to keep our results comparable to others achieved using the SimpleQuestions benchmark dataset.

While the LC-QuAD 2.0 benchmark dataset did not provide URIs to its questions, it provided sample solution queries for both Wikidata, and DBpedia. However, these queries use Wikidata's identifiers for both KGs. Take for example the question "What is the foundational document of the Soviet Union?". Listing 7 shows the sample solution query for Wikidata, while listing 8 shows the sample solution query for DBpedia.

```
select distinct ?obj where {
    wd:Q15180 wdt:P457 ?obj .
    ?obj wdt:P31 wd:Q49848
}
```

Listing 7: SPARQL-query for Wikidata to answer the question "What is the foundational document of the Soviet Union?" in the LCQuAD 2.0 benchmark.

```
select distinct ?obj where {
    ?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject> <
        http://wikidata.dbpedia.org/resource/Q15180> .
    ?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
         <http://www.wikidata.org/entity/P457> .
    ?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> ?
        obj .
    ?obj <http://www.wikidata.org/entity/P31> <http://wikidata.
        dbpedia.org/resource/Q49848>
}
```

Listing 8: SPARQL-query for DBpedia to answer the question "What is the foundational document of the Soviet Union?" in the LCQuAD 2.0 benchmark.

For this reason, it was possible for us to single out the sample queries' individual URIs for Wikidata in the same way as we did for the SimpleQuestions benchmark dataset and used them for both Wikidata and the DBpedia in our study.

Last, since the StudentQuestions and the WDAquaCore0Questions benchmark datasets did come with missing or faulty sample solution queries, we manually added the corresponding URIs for each question.

4.1.2 Benchmark Analysis

Upon analyzing the selected benchmarks, we noted a few problems affecting their reliability.

This section is therefore aimed ad giving a summary of these problems.

Differences between Wikidata and DBpedia

While both versions of the LC-QuAD 2.0 benchmark came with a sample solution query for each question, we observed a difference between the results for one of our sample questions. For the question "How many country citizenship are taken by Antonio José de Sucre Farell?" Wikidata's sample solution query returned two, while its DBpedia counterpart returned zero. In order to evaluate the results in this study, we followed the sample solution queries and marked only those queries correct which returned the same answer as the sample solution query for the same underling KG. Listing 9 shows the sample solution query for Wikidata, while listing 10 shows its DBpedia counterpart.

```
SELECT (COUNT(?obj) AS ?value ) {
    wd:Q189779 wdt:P27 ?obj
}
```

Listing 9: Sample solution query for the question "How many country citizenship are taken by Antonio José de Sucre Farell?" for Wikidata.

```
SELECT (COUNT(?obj) AS ?objs ) {
    ?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject> <
       http://wikidata.dbpedia.org/resource/Q189779> .
    ?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate>
        <http://www.wikidata.org/entity/P27> .
    ?statement <http://www.w3.org/1999/02/22-rdf-syntax-ns#object> ?
       obj .
}
```

Listing 10: Sample solution query for the question "How many country citizenship are taken by Antonio José de Sucre Farell?" for DBpedia.

SimpleQuestions Labels

Despite the simplicity of its questions' structure, the SimpleQuestions benchmark dataset for Wikidata came with two problems.

First, some of the predicate-identifiers have been labeled incorrectly. Take for example the question "what book was written by rabindranath tagore?". This question's corresponding predicate-identifier provided by the benchmark dataset is *R50*. However, no identifiers starting with an R exist for Wikidata. When replacing the R with a P the resulting identifier represents the property "author" which does not only make sense in the context of the question but would, when used to construct a query in combination with the subject-identifier provided for this question, lead to the correct answer to this question.

Second, some of the questions relate barely or not at all to their corresponding triple. As an example, the question "what coast does was nominated for Classical Album of the Year" does not only make no sense from a syntactic and semantic point of view but its triple <Q3287722, P27, Q1008> which represents the resources "Marc-Éric Gueí", "country of citizenship" and "Ivory Coast" indicates that the actual question related to this triple more likely aims at asking either for Marc-Éric Gueí's country of citizenship or for people whose country of citizenship is Ivory Coast.

WDAquaCore0Question

Upon taking a closer look at the WDAquaCore0Question's questions and their corresponding sample solution queries, it became clear that also the benchmark was not free of potential problems.

The first notable observation relates to the formulation of the questions. While some questions in the dataset, such as the question, "who won the eurovision song contest 2016" do not seem particularly poorly formulated, other questions do. Take for example the question "who leonardo dicaprio" which is missing all stop words and punctuation, and looks like a question that has been reduced to its keywords. While a properly formulated version of this question such as "Who is Leonardo Dicaprio?" can still be assumed somewhat confidently, its solution remains ambiguously. From this formulation, it is impossible to argue for example whether the answer "An American

actor." or "A (at the time of writing) 48-year-old man." present a better fitting or more correct answer to the question. Therefore, the sample solution query for this question, as shown in listing 11, which returns the Q-identifier for Leonardo DiCaprio's Wikidata entry, can in theory not be used to assess the correctness of the result of a query generated by QA-system that would for example return the term "man" and Leonardo DiCaprio's age.

```
SELECT DISTINCT ?x WHERE {
    VALUES ?x {
        <http://www.wikidata.org/entity/Q38111>
    }
    <http://www.wikidata.org/entity/Q38111> ?p <http://www.wikidata.
        org/entity/Q5>
} limit 1000
```

Listing 11: Sample solution of a SPARQL-query answering the question "who leonardo dicaprio" in the WDAquaCore0Questions benchmark.

Other questions consist only of single terms, such as "lyon" or "london". These can therefore hardly be considered questions, and the problem of ambiguity wrt. their answer worsens. Without an interrogative word such as "what" or a word providing context such as "city" a question, such "london" becomes so vague that basically all samples solution queries become unusable to evaluate the correctness of other queries' results.

The next problem lies within the labeling of the questions. The WDAquaCore0 QA-system was able to answer questions written in four languages over Wikidata [18]. Therefore, the questions in the WDAquaCore0Questions dataset are labeled according to their language. However, for some questions these labels are incorrect. Take for example the questions "tagesschau website" or "zell teilung". "Tagesschau" is a German television news service [65]. Since the term website exists in both the English and the German language ("Website") this question can be labeled as both English, and German. The phrase "Zell Teilung" however, was probably meant to be term "Zellteilung", which is the German term for cell division and does not exist in the English language. Yet, this question has been labeled as English.

Continuing to look at the question "tagesschau website" brings us to the last problem of the dataset, namely queries whose results can almost confidently be categorized as wrong. While the proper solution to the question "tagesschau website" is again ambiguous, the provided sample solution query shown in listing 12 returns Wikidata's Q-identifier for "Zell im Fichtelgebirge" which is a German market municipality located in Bavaria and has obviously close to nothing to do with cell division and rather stems from the term "zell".

```
SELECT ?x where {
    VALUES ?x {
        <http://www.wikidata.org/entity/Q31614>
    }
}
```

Listing 12: Sample solution of a SPARQL-query answering the question "zell teilung" in the WDAquaCore0Questions benchmark.

Note that we used the term "almost confidently" here, since this answer could still be considered correct if the website would mention this market municipality or was hosted in it. However, one would most likely still assess this answer as incorrect in this case, since no additional information that specifies this context is part of the answer.

Since the benchmark's GitHub-page did not specify further whether the sample solution queries stem from users or have been refined in any way [14] the problems observed here lead us to believe that these queries stem directly from the WDAquaCore0 QA-system itself.

4.2 Implementation of the Experiment

To limit the influences of human error in our experiment, we used the R programming language to conduct this study. While this does not only allow us to reliably execute each step in our study and to keep the results reproducible, it also enables us to automate parts of the evaluation process, as well as to refine and extend this experiment's setup if needed.

Additionally, to the default toolkit of R, the "openai" library for R was used as the endpoint to OpenAI's API [52] while Wikidata's query service for R was used to execute all SPARQL queries on Wikidata and retrieve their results [46].

Since this study aims not only at evaluating a single but two different GPT models over a multitude of different benchmark datasets for different KGs we set up our R-scripts to allow for flexibility wrt. these aspects. Listing 13 shows the corresponding R-code used to set up the evaluation of the assessed component.

```
#### Setup ----
# Knowledge graph
cKG <- "Wikidata"
# Query language
cQL <- "SPARQL"
```

```
# openAI API key
cAPIKey <- "sk-..."
# Temperature
nTemp <- 0
# GPT version
cGPTVersion <- "gpt-3.5-turbo"
# Benchmark path
cBMPath <- "C:/Users/..."
# Result path
cResPath <- "C:/Users/..."
# Benchmark
cBenchmark <- "QALD_9"
# Component
cComponent <- "Phrase_Variation"
```

Listing 13: R-code to set up the evaluation of the GPT models.

The variable *cKG* defines the KG used and is used not only for naming the resulting files but also for the generation of the given prompts used.

Despite limiting this study to KGs that can be queried using SPARQL and therefore SPARQL being the only query language, we used the variable *cQL* to enable our study to be easily extended to different query languages.

The variables *cAPIKey* and *cGPTVersion* are used to enable and select the GPT-model used in this study.

cTemp sets the so-called "temperature" parameter of the given GPT model which allows controlling how deterministic the behavior of the model should behave with a value of 0 resulting in the model behaving completely deterministic.

On the other hand, *cResPath* and *cComponent* are used to name and store the results for each iteration of each component's assessment.

Last, the variable *cBMPath* sets the path to load the benchmark sample datasets from and *cBenchmark* defines witch benchmark should be loaded for the given iteration of the assessment. Listing 14 shows the R-code used to load the benchmark sample datasets.

```
#### Benchmark ----
dfBenchmark <- readRDS(
  paste0(
    cBMPath,
    "SampleSet_",
    cBenchmark,
    ".rds"
  )
)
```

Listing 14: R-code to load the given benchmark sample dataset.

Furthermore, we recorded the time it took for each iteration of each KGQA-component's assessment to complete. This was done to get an idea on how efficient the GPT-models are when used in KGQA and to assess whether they provide a reliable option for KGQA-tasks when trying to answer a larger number of questions. Listing 15 shows the R-code used to record this execution time, where $lKGQA\_Component$ is a list object storing the results of the given KGQA-task.

```
lKGQA_Component <- list()
tStart <- Sys.time()
.
.
.
tEnd <- Sys.time()
lKGQA_Component[[length(lKGQA_Component) + 1]] <- tEnd -
   tStart
```

Listing 15: R-code to record the execution time for the assesment of a KGQA-task over a given benchmark.

Finally, all the results have been stored as RDS-files to be easily imported and analyzed using the R programming language.

4.3 GPT-Based Components of KGQA

In order to determine the GPT-models' fitness for KGQA we first determined individual components or tasks commonly occurring in KGQA before developing methods to let these tasks be executed by the GPT-models. Since many tasks occurring in KGQA require for example specific knowledge of the underlying graph, we excluded such tasks from our analysis and limited this study to tasks that can be executed solely by the GPT-models and some additional information provided within the given prompts.

To structure our findings, we follow the afore-mentioned categorization of different KGQA-tasks proposed by Diefenbach et al. [17].

4.3.1 Question Analysis

In order to evaluate the GPT-models' fits for "Question Analysis" tasks, we tried to use the LLMs analyzed in this study to execute the tasks of phrase variation, question variation, question pseudonymization, type detection, POS tagging, named entity recognition, entity and relationship detection and semantic parsing.

Phrase Variation
Phrase variation describes the task of generating variations of individual phrases or words within a natural language question. The aim of this task is to increase the chance of finding the correct resources for a given question within a given KG by increasing the number of phrases for which potential resources can be found. An established KGQA-system performing a task that follows this definition of phrase variation is [67].

For example, the phrase “male” could be varied by instead using the phrase “man” or the phrase “guy”.

To assess the GPT-models’ performance, wrt. this task, we first asked the LLMs to determine all possible terms or phrases within the question before asking the models to generate 3 variations for each term or phrase. We decided to only generate 3 variations to limit the number of results that need to be evaluated while simultaneously enabling a detailed enough assessment of the models’ performance wrt. this task. Listings listings 16 and 17 show the prompts used to determine and vary the phrases occurring in the natural language question.

```
Please give me all possible terms/phrases of the question '
   cQuestion', without stop words separated by a ;
```

Listing 16: Prompt used to determine all possible phrases in a natural language question.

```
Please give me nVariations variations/synonyms for each term
   and phrase in the question 'cPhrase', without the original
   term, without numbering, in one line and separated by a ;
```

Listing 17: Prompt used to vary a phrase.

Here the variable *cQuestion* represents the natural language question while *cPhrase* represents the given term or phrase. *nVariations* on the other hand, defines the number of variations to be created which as mentioned earlier has been set to 3 across the entire study.

Additional orders such as “without stop words” have been given to the LLMs to enable further automated processing of the results within R. Listing 18 shows the entire R-code used for the phrase variation task.

```
#### Phrase Variation ----
lPhrase_Variation <- list()
tStart <- Sys.time()
for(q in 1:nrow(dfBenchmark)){
  # Phrase
  cQuestion <- dfBenchmark$Question[q]
  # Number of variations
  nVariations <- 3

  # Prompt 1
  cPrompt <- paste0(
    "Please give me all possible terms/phrases of the
       question '",
```

```
  cQuestion,
  "', without stop words separated by a ;"
)

# Result 1 - Phrases
lPhrases <- create_chat_completion(
  messages = list(
    list(
      "role" = "user",
      "content" = cPrompt
    )
  ),
  model = cGPTVersion,
  openai_api_key = cAPIKey,
  temperature = nTemp
)
lPhrases <- lPhrases$choices$message.content
lPhrases <- unlist(lPhrases)
lPhrases <- unlist(strsplit(lPhrases, ";"))

### Phrase variation
dfVariations <- data.frame(Phrase = NA, Variations = NA
   )
for(i in seq_along(lPhrases)){
  # Selecting the phrase
  cPhrase <- trimws(lPhrases[i])

  # Prompt 2
  cPrompt <- paste0(
    "Please give me",
    nVariations,
    " variations/synonyms for each term and phrase in
       the question '",
    cPhrase,
    "', without the original term, without numbering,
       in one line and separated by a ;"
  )

  # Result 2 - Variations
  lRes <- create_chat_completion(
    messages = list(
      list(
        "role" = "user",
```

```
            "content" = cPrompt
          )
        ),
        model = cGPTVersion,
        openai_api_key = cAPIKey,
        temperature = nTemp
      )
      lRes <- lRes$choices$message.content
      lRes <- trimws(unlist(strsplit(lRes, ";")))
      lRes <- lRes[1:nVariations]
      lRes <- gsub("^(.*?): ", "", lRes)

      dfVariations[i, 1] <- cPhrase
      dfVariations[i, 2] <- paste(lRes, sep = "", collapse
         = ", ")
    }
    lPhrase_Variation[[q]] <- dfVariations
}
tEnd <- Sys.time()
lPhrase_Variation[[length(lPhrase_Variation) + 1]] <-
   tEnd - tStart

### Saving
saveRDS(
  lPhrase_Variation,
  file = paste0(
    cResPath,
    cComponent,
    "_",
    cBenchmark,
    "_",
    cKG,
    ".rds"
  )
)
```

Listing 18: R-code used for Phrase Variation

The R-codes used to execute the assessment of most of the following tasks in the "Question Analysis" step are mostly identical to the one used during for the assessment of the phrase variation task and will therefore not be shown.

Question Variation

While phrase variation deals with the generation of different variations for

a single phrase or word in a natural language question the task of question variation describes the generation of variations of the whole natural language question. Therefore, the goal of this task lies similar to the phrase variation task in increasing the likelihood of finding the proper resources in the KG and therefore being able to generate the right queries to answer the natural language question by increasing the number of natural language questions leading to a larger number of potential results. See [67] for a KGQA-system incorporating question variation.

Similar to the example for phrase variation, an example for question variation would be to change the question "Is Joel Zimmerman male or female?" to "What is the gender of Joel Zimmerman?",

We assessed the LLMs' performance wrt. the question variation task in a similar way as we did for the phrase variation task. For each question 3 variations were created. The prompt used for this task is shown in listing 19.

```
Please rephrase the question 'cQuestion', nVariations times,
   without numbering, in one line and separated by a ;
```

Listing 19: Prompt used to generate variations of a natural language question.

Question Pseudonymization

We define question pseudonymization as the translation of a question into a pseudo-code like version of itself. Here the idea is to reduce the question to a form that unambiguously represents its true intention using the minimal amount of words and characters necessary. Such a minimal form might enable different tasks in the KGQA process such as phrase variation or entity and relationship detection to yield better results than when using the full natural language question.

Following our previous example the question "What is the gender of Joel Zimmerman?" could be pseudonymized to the phrase "Gender Joel Zimmerman".

To achieve this task using the GPT-models we used the prompt shown in listing 20.

```
Please give me a simplified pseudo-language version of the
   question 'cQuestion', without comments and neglecting stop
   words.
```

Listing 20: Prompt used to generate a pseudo-code like version of a natural

language question.

Type Detection
Type detection refers to the categorization of a given natural language question wrt. to a set of question types. In order to asses the GPT-models' performance wrt. this task we followed Orogat et al. and the question types defined in the CBench benchmarking suite [43]. These question types do not only provide a generally good categorization scheme for questions but using them allows for a comparison of the GPT-models' results against an established approach, in this case CBench. Type detection is used in various KGQA-systems such as [60, 33].

To determine the question types of the natural language questions in our benchmark sample datasets we first defined a string containing all question types as shown in listing 21 before generating a prompt that asks the GPT-model to decide to which of these question types the given natural language question would most likely correspond to. This prompt is shown in listing 22.

```
#### Type Detection ----
# Types
cTypes <- c("What", "When", "Where", "Which", "Who", "
   Whom", "Whose", "How",
            "Yes/No", "Requests", "Topical")
cTypes <- paste(cTypes, sep = "", collapse = "', '")
cTypes <- paste0("'", cTypes, "'")
```

Listing 21: R-code used to define the question types following CBench.

```
Which of the question types 'What', 'When', 'Where', 'Which', '
   Who', 'Whom', 'Whose', 'How', 'Yes/No', 'Requests', '
   Topical' best describes the question 'cQuestion'? Please
   give me only the type word!
```

Listing 22: Prompt used to determine the type of a natural language question.

Following this definition, our example question "What is the gender of Joel Zimmerman?" would be classified as a "What"-question.

POS Tagging
Part-of-speech (POS) tagging defines the process of labeling each word in a sentence with its corresponding part of speech, such as noun, verb, adjective,

etc., and is an essential task in the context of pre-processing for NLP applications, such as text classification [35]. In the context of KGQA, POS tagging aims at disambiguating the meaning of a question and to therefore increase the effectiveness of the KGQA process' result. POS tagging can be found in various KGQA systems, such as for example [54]. While both handmade rule based and stochastic method based POS tagging algorithms are commonly used we did supply the GPT models with any rules neither did we specify which of these methods the models should use in this study. The prompt used to execute the POS tagging task via the GPT models is shown in listing 23.

```
Please do part-of-speech tagging for the question 'cQuestion'
   without adding comments in the format: word - tag
```

Listing 23: Prompt used for POS tagging.

Note that the section "without adding comments in the format: word - tag" was added in order to easily store and analyze the results in the form of a table.

Our example question "What is the gender of Joel Zimmerman?" would be tagged as follows, "gender", "Joe" and "Zimmerman" would be tagged as nouns, "What" would be tagged as pronoun, "is" would be tagged as a verb, "the" would be tagged as a determiner, and lastly "of" would be tagged as a preposition.

Named Entity Recognition

The term named entity recognition describes the task of locating and categorizing important nouns and proper nouns, mainly contiguous spans of tokens referring to a resource in a text [39]. In KGQA systems such as [16, 33, 13, 54, 28] named entity recognition tools are used to identify continuous spans of tokens within a question that refer to a resource in the given KG.

In our example case, the term "Joe Zimmerman" should be recognized as one name entity while it could be recognized as the two entities "Joel" and "Zimmerman".

To test the GPT models' fit for this task the prompt shown in listing 24 was used.

```
Please give me all named entities in the question 'cQuestion'
   without adding comments
```

Listing 24: Prompt used to recognize named entities.

Entity and Relationship Detection
Since the task of recognizing named entities is usually strongly dependent on knowledge about the given domain [17] we adapted this task by letting the GPT models not only detect a question's named entities but also the relationships between them. In theory, this should provide further components along the KGQA pipeline with a better understanding of the questions intention and therefore could result in increased performance of the KGQA system. This task will be referred to as entity relationship detection from here on and is found in systems, such as [60, 33, 54, 28].

The prompt used to detect both a question's named entities and their relationships is shown in listing 25.

```
Please give me all named entities and their relations for the
   question 'cQuestion' and return the results in 2 lines: the
    entities and the relations
```

Listing 25: Prompt used to recognize named entities and their relations.

To follow our example question, the entities "gender" and "Joel Zimmerman" are related over the term "of" (i.e., gender of Joe Zimmerman).

Semantic Parsing
"Semantic Parsing" defines the process of coupling syntactic rules to a text's semantic composition to generate a semantic interpretation (or logical form) of the text [17]. It finds its application various KGQA-systems, such as [60, 33, 28].

Staying with Joel Zimmerman the phrase "Joel Zimmerman is male" could be parsed as follows. "Joel Zimmerman" would be categorized as a noun phrase *(NP)*. Similarly, "male" could be categorized as a noun phrase. With these categorizations, "is" could be categorized as $(S\backslash NP)/(S\backslash NP)$, meaning it can be combined with a noun phrase on both the left and the right side to form a sentence S.

To test how the GPT-models would handle this task we did not supply them with any particular grammars or other rules. Rather we aimed at keeping the prompt as minimalistic and simple as possible, and focused on the structure and format of the results. The addition " and return the results

in 3 lines: the logical form, the entities and the actions and do not add any other linebreaks but separate the individual entities and actions by a comma" has been made to format the results of this steps in a way that allows for easier analysis. Listing 26 shows the prompt used for "Semantic Parsing".

```
Please apply semantic parsing to the question 'cQuestion' and
   return the results in 3 lines: the logical form, the
   entities and the actions and do not add any other
   linebreaks but separate the indivudal entites and actions
   by a comma
```

Listing 26: Prompt used for semantic parsing.

4.3.2 Phrase Mapping

Under "Phrase Mapping" we classify all tasks that deal with linking the elements of a natural language question such as words and phrases to resource identifiers in a given KG. While various approaches exist to handle this task, many of them require knowledge about the underlying KG that goes beyond what can be reasonably supplied to a GPT-model via its prompt. We therefore limit this study to approaches in which no specific knowledge of the underlying KG is assumed or supplied.

Direct Phrase Mapping We label the first variation of the "Phrase Mapping" task as direct phrase mapping. In this approach we tried to generate the corresponding resource identifier of a given KG for each phrase or keyword in a natural language question. Almost all KGQA-systems link individual phrases and keywords to a KG's resource identifiers.

In order to use the GPT-models to execute this task we used the keywords that have either been supplied by the selected benchmark or the ones that have been manually created by us when preparing the benchmark sample datasets.

Listing 27 shows the R-code used to generate the GPT-models' results for the direct phrase mapping task where we first split up the given phrases for each question before generating their corresponding resource identifiers. These identifiers have been combined and stored afterwards.

```
#### Direct Phrase Mapping ----
lDirect_Phrase_mapping <- list()
tStart <- Sys.time()
for(q in 1:nrow(dfBenchmark)){
```

```
  cPhrases <- dfBenchmark$Keywords[q]
  cPhrases <- unlist(strsplit(cPhrases, ", "))

  dfRIs <- data.frame(Phrase = NA, Resource_Identifier =
     NA)
  # Phrase
  for(p in seq_along(cPhrases)){
    cPhrase <- cPhrases[p]

    # Prompt
    cPrompt <- paste0(
      "Please give me ",
      cKG,
      "'s resource identifier for the term/phrase '",
      cPhrase,
      "' without adding any comments"
    )

    # Result
    lRes <- create_chat_completion(
      messages = list(
        list(
          "role" = "user",
          "content" = cPrompt
        )
      ),
      model = cGPTVersion,
      openai_api_key = cAPIKey,
      temperature = nTemp
    )
    lRes <- lRes$choices$message.content

    cMappedPhrase <- c(cPhrase, lRes)
    dfRIs <- rbind(dfRIs, cMappedPhrase)
    colnames(dfRIs) <- c("Phrase", "Resource_Identifier")
  }
  dfRIs <- dfRIs[2:nrow(dfRIs), ]
  lDirect_Phrase_mapping[[q]] <- dfRIs
}
tEnd <- Sys.time()
lDirect_Phrase_mapping[[length(lDirect_Phrase_mapping) +
   1]] <- tEnd - tStart
```

Listing 27: R-code use to assess the GPT-models' performance wrt. direct

phrase mapping.

To generate these resource identifiers the prompt shown in listing 28 has been used, where the variable cKG refers to the underlying KG, and where $cPhrase$ refers to the given phrase or keyword.

```
Please give me 'cKG''s resource identifier for the term/phrase
    'cPhrase' without adding any comments
```

Listing 28: Prompt used to execute direct phrase mapping using the GPT-models.

Again, cKG refers to the underlying KG, while $cPhrase$ represents the phrase or term for which the corresponding resource identifier should be generated.

In order to analyze the resource identifiers for Wikidata generated by the models, we retrieved their labels using the query shown in listing 29, where $cQID$ represents the resource identifier generated by the LLM.

```
SELECT ?name
WHERE {
    wd:cQID rdfs:label ?name.
    FILTER(LANG(?name)="en").
}
```

Listing 29: Query used to retrieve resource identifier labels from Wikidata.

Global Phrase Mapping When trying to map individual phrases or keywords to a resource identifier in a given KG, some potential problems arise. First, the meaning of a given phrase might be strongly dependent on the context and therefore the entire question. Second, individual keywords or phrases might not be directly linkable to a certain resource in the underlying KG, leading to the return of either no result at all or linking the closest resource to the given keyword and phrase. These problems could lead to the mapped resource identifiers being either useless or even harmful to the generation of a functioning SPARQL query.

For this reason, we define global phrase mapping as the act of linking an underlying KG's corresponding resource identifiers to a given natural language question based on the entire question and therefore without isolating individual phrases or keywords.

In order to generate the resource identifiers that correspond to the question, we first used the prompt shown in listing 30.

```
Please give me all of DBpedia's relevant resource identifiers
   for the question 'Who is the daughter of Ingrid Bergman
   married to?' without adding any comments
```

Listing 30: First prompt used to execute global phrase mapping using the GPT-models.

However, using this prompt resulted in the generation of only one resource identifier for each question during our first trial using GPT-3.5. Take for example the question "Who is the daughter of Ingrid Bergman married to?" for which GPT-3.5 returned only the resource identifier "http://dbpedia.org/resource/Isabella_Rossellini" for DBpedia.
To enable the generation of multiple resource identifiers, we rephrased the original prompt, resulting in the prompt shown in listing 31.

```
Please give me all resource identifiers for DBpedia that relate
    to the question 'How often did Jane Fonda marry?' without
   adding any comments. Limit the number of resource
   identifiers per term to 10.
```

Listing 31: Second prompt used to execute global phrase mapping using the GPT-models.

Using the same question as in the previous example, GPT-3.5 now returns the resource identifiers "http://dbpedia.org/page/Ingrid_Bergman", "http://dbpedia.org/ontology/spouse", "http://dbpedia.org/page/Roberto_Rossellini" and "http://dbpedia.org/page/Isabella_Rossellini".

Note that the phrase "Limit the number of resource identifiers per term to 10." was added to the prompt since GPT-3.5 would otherwise sometimes return an unreasonably large number of resource identifiers, from which most of them have nothing to do with the original question.

4.3.3 Disambiguation

Diefenbach et al. define the "Disambiguation" task as the task of identifying which of the resources found in the "Phrase Mapping" task is the right one [17]. The techniques used for this task range from using local disambiguation and graph search techniques to statistical approaches such as the usage of hidden Markov models (HMM), integer linear programming (ILP) and Markov logic networks (MLN) and even the incorporation of user feedback.

Imagine we found the possible identifiers for the term "bark" for the question "Which dog does bark the loudest?" during the phrase mapping task. We might have found the identifier for a tree's out layer (i.e. its bark) as well as the sound a dog makes. Given the context of the question and the structure of the underlying KG, we can determine that for our question, we most likely are interested in the second identifier.

While it would be possible to test LLMs wrt. this task it would require the generation of a set of multiple resources for each term or phrase in all the questions used in this study. Each set of resources should additionally contain at least one resource fit and one resource unfit to represent the given term or phrase. This generation of sets of resources would require significant effort and might be reliant on the usage of the component of different KGQA systems, which lies outside the scope of this study. We therefore did not test the LLMs' fit for this task.

4.3.4 Query Construction

In the process of constructing the query to answer a given question, KGQA-systems must deal with the semantic gap problem, which refers to the fact that a KGQA-system might encode an information differently from what one could deduce from the question [17]. Approaches in this field range from using templates or semantic information resulting from the "Question Analysis"-phase to ones using machine learning or using no SPARQL at all. Again, we limited the analyzed approaches to those that mainly rely on the GPT-models instead of ones requiring results from previous KGQA-components.

Direct Construction Direct construction labels the simplest approach analyzed in this study. This approach is in line with our preliminary experiment in which GPT-3.5 was used to directly generate SPARQL-queries to answer a given question over Wikidata without the aid of additional information, such as resource identifiers [31].

To enable this approach to be used across multiple KGs and query languages, the prompt shown in listing 32 was used.

```
Please write me a cQL-query without comments for cKG to answer
   the question 'cQuestion'
```

Listing 32: Prompt used for direct query construction.

To evaluate the generated queries we followed the approach used in our initial study in which retrieved the results for each query and assign the queries to

three categories [31]. *Executable* queries label queries that are syntactically correct, i.e., queries that can be executed without receiving an error message. Since the syntactical correctness of a query does not allow one to make statements about its fit to answer a certain question, we label queries that can be executed but do not return any result as *empty* queries while queries whose results were considered to be a possible correct answer to the given question as *correct* queries.

URI-Fed Construction Since we already showed that the GPT-models are not capable of reliably linking phrases or terms to the resources with a given KG, one cannot expect the LLMs to construct fitting SPARQL-queries without being fed additional information wrt. the necessary resource identifiers. For this reason, the direct construction task has been modified by supplying the model with the resource identifiers taken from the sample solution queries. This approach to construct queries was labeled URI-fed construction.

To test the GPT-models for this version of the query construction task, we added the phrase "using the URIs 'cURIs"' to the original prompt as shown in listing 33. Here, *cURIs* represents a string containing both the KGs resource identifiers, and their labels. An example of this string for the question "What is the foundational document of the Soviet Union?" in the LC-QuAD 2.0 benchmark for Wikidata would be "Q15180 = Soviet Union, Q49848 = document, P457 = foundational text, P31 = instance of".

```
Please write me a cQL-query without comments for cKG to answer
   the question'cQuestion' using the URIs 'cURIs'
```

Listing 33: Prompt used for direct query construction.

Template-Based Construction While the direct construction and URI fed construction tasks represent versions of the query construction task that solely rely on an LLM's knowledge about SPARQL, another common approach is to use SPARQL-templates to construct a question's corresponding SPARQL-query. These templates based approaches usually make use of so-called slots. A slot represents a triple consisting of the natural language expression in the question, the type of the resource identifier (class, property or resource) and a placeholder variable. Upon linking the question to a given SPARQL template, these slots will then be replaced by the resource identifiers corresponding to the question [58].

In order to test the LLMs for this task, we followed Park et al. [44] and the SPARQL-templates used by them. Listing 34 shows these templates.

```
# Boolean
ASK WHERE { ?x ?p ?y. }
# Simple
SELECT DISTINCT ?x WHERE { ?x ?p ?y . (Option:?x rdf:type ?c.)}
# Count
SELECT COUNT (DISTINCT ?x) WHERE { ?x ?p ?y. }
# Filter
SELECT DISTINCT ?x WHERE {?x ?p ?y. } ORDER BY DESC(?x) OFFSET 0
    LIMIT n
# Order
SELECT DISTINCT ?x WHERE {?x ?p ?y . FILTER (?y <1950)}
```

Listing 34: SPARQL-templates used in the template based-construction task.

Each template corresponds to a certain type of question, namely boolean, count, filter, order and simple questions. In our approach, the LLM first links a question to one of the question types. For this we supplied the LMM with both the question types and their definitions as seen in listing 36. Due to the lack of a proper definition of a simple question in the original paper by Park et al. [58] we used the definition of Yani and Krisnadhi [66].

```
Which of the following question-types best fits the question '
    cQuestion'?
Boolean: Shall be answered with a yes/no or true/false
    statement.\n",
Count: The answer is based around a count of something (e.g.,
    questions starting with 'How many..').
Filter: The question contains a comparative (e.g., 'more
    citizens than..').
Order: The question contains a superlative (e.g., 'most
    citizens').\n",
Simple: The answer can be captured by a factoid statement with
    one relation or predicate.
Please return only the question type!
```

Listing 35: Prompt used to determine the question type in the template-based construction task.

After receiving the question type selected by the LLM, we expanded the prompt used in the direct construction task by instructing the model to make use of the SPARQL-template corresponding to the detected question type.

```
Please write me a cQL-query without comments for cKG to answer
   the question 'cQuestion' using the following cQL-template:
   'cTemplate'
```

Listing 36: Prompt used to generate SPARQL-queries in the template-based construction task.

URI-Fed Template-Based Construction Last, we also extended the template-based construction task by supplying the LLMs with the correct resource identifiers. This was done in the same way as for the URI-fed construction task. Listing 37 shows the resulting prompt used to generate the SPARQL-queries in the URI-fed template-based query construction task.

```
Please write me a cQL-query without comments for cKG to answer
   the question 'cQuestion' using the following cQL-template:
   'cTemplate' and the URIs 'cURIs'
```

Listing 37: Prompt used for direct query construction.

4.3.5 Querying Distributed Knowledge

Querying Distributed Knowledge refers to techniques that answer a single question over multiple KGs by either assuming the KGs to be disjoint graphs or that the KGs are interlinked [17].

However, due to this study's focus on LLMs and their performance, wrt. KGQA we did limit our area of interest to answering questions over a single KG.

5 Results

Given the variety of different KGQA-tasks analyzed in the study, we also received a variety of results. The quality of these results does not only differ between the tasks themselves, but also between the LLMs used.

This section is dedicated to presenting the acquired results.

5.1 Question Analysis

Despite the fact that the LLMs have generally been able to perform many of the tasks in the "Question Analysis" phase well, various problems became apparent. This section therefore presents these results.

5.1.1 Phrase Variation

While GPT-3.5 was able to generate reasonable variations for most phrases or keywords in all the sample questions, a few problems and habits were observed repeatedly.

First, some of the variations deviate in meaning from the original phrase or keyword. Take for example the term "island" in the question "On which island is the HQ location of the Carlsberg Group" in the LC-QuAD 2.0 benchmark dataset. One of the variations generated by GPT-3.5 was "atoll" which while being defined as an island is also defined by its ring-like shape and the inclusion of a coral ring encircling a lagoon [64]. If used, this variation therefore would strongly limit the possible instances of the original object that is an island.

A similar issue is the observed loss of context. In the question "What is the architecture firm that is based in Saint Longinus" in the LC-QuAD 2.0 benchmark, the term "based" spawns the variations "derived", "grounded" and "build upon" while the term "taken" in the question "How many country citizenship are taken by Antonio José de Sucre Farell" spawns the variations "captured", "derived", and "acquired". Obviously, most of these variations might be synonyms for the original terms, but do not hold the same meaning in the context of the full questions.

On the other hand, some of the variations are only remotely related to the original phrase or keyword. For example, the term "baseball" in the LC-QuAD 2.0 benchmark's question "Which is the international sport governing body for authority of baseball?" spawned the variations "America's pastime", "ballgame" and "slugger". The term "slugger" defines "a hard-hitting batter in baseball" [37] and therefore not the game itself.

Obvious patterns can be observed wrt. names and dates. While names such as "Simon Baron-Cohen" in the LC-QuAD 2.0 benchmark datasets' question "What is Loop ID for Simon Baron-Cohen?" usually get shorter in their variations ("Simon Baron-Cohen", "Simon B. Cohen" and "S. Baron-Cohen") dates written as number will be written out. For example, "2023" in the StudentQuestions benchmark datasets' question "Where does the handball world cup take place this year (2023)?" spawned the variations "twenty twenty-three", "two thousand twenty-three" and "variations".

The last term already points out additional problems in the form of random results. These often include parts of the prompts as the term "lyon" from the WDAquaCore0Questions benchmark datasets' question "lyon" spawns the variations "variations", "synonyms", "phrases", "possible terms", "one line", "separated", "by a" and two empty results.
Last, the LLM sometimes returned more variations than required. While

we asked for 3 variations per phrase, GPT-3.5 generated 9 variations for the term "genre" in the LC-QuAD 2.0 benchmark datasets' question "who is the record label and genre of The_Velvet_Underground?" ("variation", "synonym", "phrase", "category", "style", "type", "kind", "classification" and "form").

The variations generated by GPT-4 were similar to those generated by GPT-3.5. While deviations in meaning of the generated phrases such as using the term "atoll" as a synonym for "island" remained, the model did comply to the given formatting instructions and consistently returned the desired number of variations for each phrase.

However, GPT-4 on average detected significantly more phrases within a question than its predecessor. For the question, "On which island is the HQ location of the Carlsberg Group" GPT-4 extracted 16 different phrases compared to the three phrases extracted by GPT-3.5.

5.1.2 Question Variation

Overall, GPT-3.5 was again able to generate reasonable variations for most of the sample questions.

Yet, poorly formulated questions (e.g., questions consisting of only one word or phrase, questions missing stop words, etc.) such as many of those found in the WDAquaCore0Questions benchmark dataset often led to unfit and rather random variations. Take the question "lyon" for example. For this question, GPT-3.5 generated the variations "What is the city of Lyon known for?", "What are some notable features of Lyon?" and "What makes Lyon unique?". As discussed earlier, the original question cannot really be defined as such, and therefore every answer related to the city of Lyon could be classified as direct. Therefore, one could make the argument that any question related to the city of Lyon could be classified as a reasonable variation of the original question. Yet, it is clear that the generated variations strongly limit the number of possible correct responses to the question.

Reasonable variations on the other hand were often limited to a single part of the question, i.e. GPT-3.5 only varies one part or phrase of the given question. See for example the question "What is the architecture firm that is based in Saint Longinus?" from the LC-QuAD 2.0 benchmark dataset for which GPT-3.5 generated the variations "What is the name of the architecture firm located in Saint Longinus?", "Which architecture firm is based in Saint Longinus?" and "What is the architecture firm situated in Saint Longinus called?".

One reoccurring habit of GPT-3.5 was changing the questions' tense.

LC-QuAD 2.0's question "What is the position of political office held by a member of Augustus' family?" spawned the variations "What political office did a member of Augustus' family hold?", "What position in politics was held by a member of Augustus' family?" and "Which political office was occupied by a member of Augustus' family?". As one can see, the generated variations are all past tense. This habit can be seen as problematic since the original question implies that a member of the Augustus' family must currently be holding a position of political office, while its variations imply that it would be enough if a member of the Augustus' family held a position of political office at any point in time.

Similar to the phrase variation task, GPT-3.5 sometimes fails at executing the task by either returning only one variation and adding parts of the prompt to it or by not following the formatting instructions. In the WDAquaCore0Questions benchmark sample dataset for example, the question "london" only spawned the variation "What are three different ways to rephrase the word 'London' in one line, separated by a semicolon?" while the question "barack obama's wife name michelle?" only spawned "What is the name of Barack Obama's wife, Michelle?\nWhat is the name of Michelle, Barack Obama's wife?\n Barack Obama's wife is named Michelle, right?"

In line with the results of the phrase variation task, GPT-4's results mostly resemble those generated by GPT-3.5. Again, the newer model shows a drastic improvement wrt. to complying with our instructions. Additionally, GPT-4 followed a more vague approach when creating variations for poorly phrased questions. For the question "Lyon" in the WDAquaCore0Questions benchmark sample dataset, GPT-4 generated the variations "What can you tell me about Lyon?", "Can you provide information on Lyon?" and "Could you elaborate on the subject of Lyon?". Compared to the variations generated by GPT-3.5 these questions allow for a much larger set of correct answers and are therefore less likely to exclude correct responses to the original question if used instead.

Last, we analyzed the lexical diversity of each question and their variations to determine whether the GPT-models have a tendency to increase or decrease the question's lexical diversity in its variations. For this, the type-token ratio (TTR) [51] has been computed for the original question, as well as for each of its variation. Afterward, the mean of each variation's TTR has been computed before averaging the results for each benchmark sample dataset. Table 6 and table 7 shows the average TTR for each benchmark sample dataset's questions and their variations generated by GPT-3.5 and GPT-4.

Table 6: Average TTR for each benchmark sample dataset using GPT-3.5.

Benchmark	Question	V1	V2	V3	Variant Mean
LC-QuAD 2.0	0.993	1	1	1	1
QALD-9	1	1	1	1	1
SimpleQuestions	0.975	0.986	0.986	1	0.991
StudentQuestions	1	1	1	1	1
WDAquaCore0Questions	1	0.980	1	1	0.993

As one can see, using GPT-3.5 the lexical diversity did on average not differ much across the different benchmark sample datasets nor between the original questions and their variations. This effect became even stronger when using GPT-4, where each variation had a lexical diversity score of one.

Table 7: Average TTR for each benchmark sample dataset using GPT-4.

Benchmark	Question	V1	V2	V3	Variant Mean
LC-QuAD 2.0	0.993	1	1	1	1
QALD-9	1	1	1	1	1
SimpleQuestions	0.975	1	1	1	1
StudentQuestions	1	1	1	1	1
WDAquaCore0Questions	1	1	1	1	1

5.1.3 Question Pseudonymization

Since no specific method to pseudonymize a question has been included in the prompt, it is hard to determine whether the GPT-models' are fit for this task. However, some observations indicate that GPT-3.5 might not prove to be reliable wrt. question pseudonymization.

First, GPT-3.5 is inconsistent with the removal of words that further specify the meaning of the question. For example, GPT-3.5 removes the word "when" from the question "When did Battle of Quiberon Bay happen?" in the LC-QuAD 2.0 benchmark sample dataset, resulting in the question's pseudonymized version "Battle Quiberon Bay happen". Without the word "when" the term "happen" in pseudonymized question could not only refer to a point in time or a time frame but also to other aspects of the battle such as what happened during it. On the other hand, the pseudonymized version of the question "How many country citizenship are taken by Antonio José de Sucre Farell?" still included the phrase "how many" and thereby keeping the original question's meaning intact.

Second, phrases indicating relations have also been removed inconsistently. So did the pseudonymized version of the question "What is the foun-

dational document of the Soviet Union?" not include the phrase "of the" while the pseudonymized version of the question did "Which is the international sport governing body for authority of baseball?" keep the phrases "..body for" and "..authority of".

Questions consisting of single terms such as those often found in the WDAquaCore0Questions benchmark dataset did not change during the question pseudonymization process.

GPT-4 seemingly took another approach to execute this task by using a type of pseudocode reminiscent to query languages, such as SQL. Here the question "When did Battle of Quiberon Bay happen?" returned "Battle Quiberon Bay date", replacing the term "happen" used by GPT-3.5 with the term "date" which is arguably better suited to capture and convey the original question's intention. Similarly to this, the question "How many country citizenship are taken by Antonio José de Sucre Farell?" now returned "Count country citizenship Antonio José de Sucre Farell", replacing the phrase "how many" with the query-like keyword "count" and therefore formulating a sort of instruction.

5.1.4 Type Detection

The results of this assessment show that the results obtained from GPT-3.5 differ vastly from those obtained by CBench. Only one of the questions in the sample datasets was labeled as request, and the question "Is the sex of anuradha sriram male or female?" has been labeled as a question that can be answered with either "yes" or "no". This label is obviously not a suitable categorization for the question, since it should be answered with either "male" or "female". The misclassification might root in a lack of description of how these individual question types are defined.

We could also observe that all answers generated by GPT-3.5 confirm to our instruction of only returning the question type label. This makes GPT-3.5's result for this task fit to be reliably processed without further intervention or i.e. in an automated manner, making it fit to be integrated in a complete KGQA-system.

While again conforming flawlessly to our instructions, the results generated by GPT-4 are not identical to those generated by its predecessor. The most notable difference occurred in the WDAquaCore0Questions benchmark sample dataset. Here GPT-4 failed to classify the question "lyon" and instead returned the message "The question 'lyon' is not a properly formed question, so it doesn't fit into any of the given question types." while GPT-3.5 classified

this question as a topical question. However, the question "london" which is arguably as little of a properly formed question as the question "lyon" was classified as a topical question by GPT-4.

Yet, GPT-4's results differed far less from the question types detected by CBench, again indicating the model's superiority over its predecessor.

A summary of the results of the CBench benchmarking suite, as well as GPT-3.5's and GPT-4's execution of the type detection task for each benchmark sample dataset, are shown in tables 8 to 10.

Table 8: Number of question types classified by CBench per benchmark sample dataset.

Type	Student	Simple	QALD-9	LC-QuAD 2.0	WDAquaCore0
What	3	9	2	9	0
When	1	0	0	1	1
Where	1	2	0	0	0
Which	2	1	1	3	2
Who	4	1	5	1	3
Whom	0	0	0	0	0
Whose	0	0	0	0	0
How	1	0	2	1	0
Yes-No	0	2	1	0	1
Requests	2	0	4	0	1
Topical	0	0	0	0	7

Table 9: Number of question types classified by GPT-3.5 per benchmark sample dataset.

Type	Student	Simple	QALD-9	LC-QuAD 2.0	WDAquaCore0
What	1	6	1	2	1
When	1	0	0	2	1
Where	1	2	0	0	1
Which	2	4	1	4	2
Who	4	1	6	1	3
Whom	0	0	0	0	0
Whose	0	0	0	0	1
How	0	0	0	0	0
Yes-No	0	1	0	0	0
Requests	0	0	1	0	0
Topical	5	1	6	6	6

Table 10: Number of question types classified by GPT-4 per benchmark sample dataset.

Type	Student	Simple	QALD-9	LC-QuAD 2.0	WDAquaCore0
What	3	9	1	8	0
When	1	0	0	2	1
Where	1	2	0	0	0
Which	2	1	1	3	2
Who	4	1	5	1	3
Whom	0	0	0	0	0
Whose	0	0	0	0	0
How	1	0	2	1	0
Yes-No	0	1	1	0	1
Requests	2	0	4	0	1
Topical	0	0	0	0	6

5.1.5 POS Tagging

Upon analyzing the results of this task, we can see that the results obtained by GPT-3.5 appear to mostly follow the Penn Treebank tagset for the English language [55]. In total 63 out of the 74 sample questions have been seemingly tagged using this tagset. Table 11 shows GPT-3.5's POS tagging result for the question "What is the foundational document of the Soviet Union?" from the LC-QuAD 2.0 benchmark sample dataset, which appears to have been tagged using the Penn Treebank tagset for the English language.

Table 11: GPT-3.5's POS tagging results for the question "What is the foundational document of the Soviet Union?".

Word	Tag	Description
What	WP	Wh-pronoun
is	VBZ	Verb, 3rd person singular present
the	DT	Determiner
foundational	JJ	Adjective
document	NN	Noun, singular or mass
of	IN	Preposition or subordinating conjunction
the	DT	Determiner
Soviet	JJ	Adjective
Union	NN	Noun, singular or mass
?	.	Sentence-final punctuation

However, for other questions, GPT-3.5 seemingly uses a different type of

tagset to tag the questions. An example for this is shown in table 12.

Table 12: GPT-3.5's POS tagging results for the question "Which is the Wikimedia category for the category of associated people of Oslo?".

Word	Tag
Which	PRON
is	VERB
the	DET
Wikimedia	NOUN
category	NOUN
for	ADP
the	DET
category	NOUN
of	ADP
associated	VERB
people	NOUN
of	ADP
Oslo	NOUN
?	PUNCT

Last, GPT-3.5 is seemingly inconsistent when it comes to the alternative labels themselves. While the labels used to tag the question shown in table 12 have all been capitalized and seemingly abbreviated, the labels shown in table 13 are presented as fully written out words in lowercase.

Table 13: GPT-3.5's POS tagging results for the question "washington square is a story by which American writer".

Word	Tag
washington	noun
square	noun
is	verb
a	determiner
story	noun
by	preposition
which	determiner
American	adjective
writer	noun

GPT-4 again used the Penn Treebank tagset for most of the questions (59 out of 74).

Yet, every question that was not tagged using the Penn Treebank tagset did follow the pattern shown in table 13, that is fully writing out the different parts of speech instead of using abbreviations.

While these results indicate that the GPT-models are in fact able to apply POS-tagging to a question, its usage of tagsets is inconsistent. Therefore, further specifying the approach that should be taken by the models such as defining which tagset should be used might enable the LLMs to reliably apply POS-tagging to questions in the process of KGQA.

5.1.6 Named Entity Recognition

While the entities recognized by GPT-3.5 mostly seemed pretty reasonable, the model again sometimes failed to follow the given formatting instruction. The question "Show me all songs from Bruce Springsteen released between 1980 and 1990." from the QALD-9 benchmark sample dataset spawned the result "Named entities in the question 'Show me all songs from Bruce Springsteen released between 1980 and 1990.' are:\n \n- Bruce Springsteen (person) \n- 1980 (date) \n- 1990 (date)".
On the other hand, for the question "lyon" in the WDAquaCore0Questions benchmark sample dataset, GPT-3.5 was not able to detect any named entities and instead returned the output "There are no named entities in the question 'lyon'." while simultaneously returning "London" as the named entity for the question "london".

In order to provide an overview of how GPT-3.5 performs, wrt. the named entity recognition task, its results for the LC-QuAD 2.0 benchmark sample dataset are shown in table 5.1.6.

Question	Named Entities
What is the foundational document of the Soviet Union?	foundational document, Soviet Union
On which island is the HQ location of the Carlsberg Group?	island, HQ, location, Carlsberg Group
What volcanic eruption occurred in the Dutch East Indies?	volcanic eruption, Dutch East Indies
What is the architecture firm that is based in Saint Longinus?	architecture firm, Saint Longinus
who is the record label and genre of The_Velvet_Underground?	The_Velvet_Underground
What is Loop ID for Simon Baron-Cohen?	Loop ID, Simon Baron-Cohen
What is the religious affiliation of the victim of the Battle of Stalingrad?	religious affiliation, victim, Battle of Stalingrad
What time did Aarhus serve as an adminstrative body at Rostock?	Aarhus, Rostock
What is the position of political office held by a member of Augustus' family?	position, political office, member, Augustus, family
What are the beliefs of the Chinese Communist Party's Chair, Hu Jintao?	beliefs, Chinese Communist Party, Chair, Hu Jintao
Which is the {international sport governing body} for {authority} of {baseball}?	- international sport governing body - authority - baseball
What are the plays of the organizer of the UMB World Three-cushion Championship?	plays, organizer, UMB World Three-cushion Championship
Which is the Wikimedia category for the category of associated people of Oslo?	Wikimedia, Oslo
When did Battle of Quiberon Bay happen?	Battle of Quiberon Bay
How many country citizenship are taken by Antonio José de Sucre Farell?	Antonio José de Sucre Farell

Again, GPT-3.5's troubles with conforming to the provided formatting

instructions or seemingly random errors in execution did not occur when using GPT-4.

Also, GPT-4 the problem of not being able to detect named entities for certain questions did occur again. This time, however, GPT-4 was only unable to detect entities in the WDAquaCore0Questions question "part of a thunderstorm".

5.1.7 Entity and Relationship Detection

GPT-3.5 was again able to provide seemingly good results, such as for the LC-QuAD 2.0 benchmark's question "How many country citizenship are taken by Antonio José de Sucre Farell?", for which it returned "Entities: Antonio José de Sucre Farell, country citizenship" and "Relations: taken by". However, most results were plagued by various problems.

Again, the model often did not comply to the given instructions, wrt. formatting its answer. To the question "Which is the international sport governing body for authority of baseball?" its answer has been returned as "Entities: \n- international sport governing body \n-authority \n-baseball \n\nRelations:\n-governing body for authority\n-authority of baseball".

To some questions, such as the question "who is the record label and genre of The_Velvet_Underground?" from the LC-QuAD 2.0 benchmark sample dataset, the answer did not contain any relationships at all.

Also, GPT-3.5 seemingly has a habit of adding descriptions or labels of some sort to some of its answers despite being asked to not add any comments. An example for this would be the question "What is Loop ID for Simon Baron-Cohen?" for which GPT-3.5 generated the answer "Loopd ID - UNKNOWN\n\nPERSON - UNKNOWN". These descriptions sometimes include named entities as well, such as in the relationship "- between (relation between 1980 and 1990)" for the question "Battle Quiberon Bay happen" from the LC-QuAD 2.0 benchmark sample dataset. Yet, named entities might be added to relationships without the addition of any descriptions as well. An example for this is the question "what is the producing company of the movie lamhe" from the SimpleQuestions benchmark sample dataset. For this question, GPT-3.5 returned "'producing company of', 'movie lamhe"' as the named entities and "" as their relationships.

However, the addition of named entities is not limited to relations but includes other named entities as well. Take for example the question "What is the foundational document of the Soviet Union?" from the LC-QuAD 2.0 benchmark sample dataset for which GPT-3.5 determined the named entities "Foundational document; Soviet Union; named entities".

Also, GPT-3.5 sometimes added brackets or quotations to its results,

which can be observed for example for the question “Who is the president of Eritrea?” from the QALD-9 benchmark sample dataset. Here, GPT-3.5 returned the entities “['president', 'Eritrea']” and the relationships “['is the president of']”.

Besides misbehavior wrt. formatting, GPT-3.5 also produced other artifacts. For some questions such as the question “What is the most recent MineCraft Java Edition version?” the model added some sort of mapping between the entities and their relationships to its output (“MineCraft Java Edition version - most recent\nEntities: MineCraft Java Edition version\nRelations: most recent”).

Last, for questions with missing relations such as those found in the WDAquaCore0Questions benchmark, GPT-3.5 might make up those relations itself. Take for example the question “tagesschau website” for which the LLM generated the named entities “tagesschau” and “website”, and the relations “tagesschau is a website”.

Using GPT-4 to detect named entities and their relationships resulted in most of the problems mentioned before not occurring at all. However, a few of the results could be assessed as wrong. In these cases, potential entities were returned as part of the relationships or relationships were returned as entities. Take for example again the question “Battle Quiberon Bay happen” for which the term “happen” was returned as an entity, or the question “what is the producing company of the movie lamhe” for which GPT-4 returned the relationship “is the producing company of”.

Despite this, most entities and relationships detected by GPT-4 were of similar quality as those returned by GPT-3.5 without any of the problems mentioned before.

These results show that GPT-4 is for the most part capable of detecting acceptable entities and relationships within a question, indicating the models' advantage over its predecessor.

5.1.8 Semantic Parsing

Upon analyzing the results, it becomes clear that GPT-3.5 did not have an easy time upon executing this task. While able to generate a logical form of some sort, a set of entities and a set of actions. The model was vastly inconsistent in how it formulated these sets.

Take for example GPT-3.5's results for the LC-QuAD 2.0 benchmark sample dataset. While the logical form for the question “What is the foundational document of the Soviet Union?” was formulated as “(document: foundational, entity: Soviet Union)”, the logical form of the question “On

which island is the HQ location of the Carlsberg Group?" reads itself as "locate_hq(Carlsberg_Group, HQ_location)". Besides the syntactical difference, the logical form lacks the action necessary to clearly state the question's intent.

This becomes even clearer when comparing the entities and actions generated for the question. The first question's entities generated by GPT-3.5 are "document" and "Soviet Union" while its actions are "is" and "foundational". Seeing this it one could assume that both actions might need to be combined (e.g, "is is foundational") in order to properly link the entities in the logical form. This approach was taken for the second question's set of actions, containing only of the action "locate_hq" which properly conveys the question's intent when combined with its entities "Carlsberg_Group" and "HQ_location".

In line with the previous tasks, GPT-3.5 also failed to properly execute the task at all times, using inconsistent notations and producing random results for certain questions, such just repeating "(and " for the question "Give me all Austrian female actors aged over 50 years that are also dancers or singers." in the StudentQuestions benchmark sample dataset.

While some of GPT-4's results resemble the ones of its predecessor, most of them follow a different pattern.

Take again the question "What is the foundational document of the Soviet Union?". While different, "HQ_Location(Carlsberg_Group, X), Island(X)", the logical form generated by GPT-4 still forms a function surrounding the entity "Carlsberg_Group". However, this time it includes a function "Island(X)" which resembles a new entity detected by the model. Additionally, the LLM also added "is" and "on which" to the previous set of actions.

While these differences are notable, the biggest difference compared to GPT-3.5 was that most of the logical forms follow a query-like structure. Take again the question "What is the foundational document of the Soviet Union?". The logical form generated by GPT-4 reads as "SELECT (foundational document) WHERE (country = 'Soviet Union')", with its corresponding entities "Foundational document" and "Soviet Union". Note that the actions corresponding to this question simply represent basic keywords found in most query languages ("SELECT, WHERE").

Despite most logical forms generated by GPT-4 following this query-like structure, its results are still too inconsistent in their form for to be graded fit for the task of "Semantic Parsing". To generate reliable results, one might include further instructions as to how the LLM should execute the task in his prompt.

5.2 Phrase Mapping

Since the tasks within the "Phrase Mapping" step of the KGQA-process differ vastly from those within the "Question Analysis" step, the results differ as well.

5.2.1 Direct Phrase Mapping

Upon analyzing its results, some patterns can be observed in GPT-3.5's behavior.

The most notable observation was made wrt. Wikidata. Here, GPT-3.5 tends to assign mostly Q-identifiers to the questions' resources while rarely using P-identifiers. Hence, GPT-3.5 assumes that most resources are represented as items rather than properties.

Next, GPT-3.5 was again inconsistent with its output's format. Some of the identifiers generated contain unwanted punctuation. Take for example the question "What time did Aarhus serve as an adminstrative body at Rostock?" from the LC-QuAD 2.0 benchmarks sample dataset. For the term "start time" GPT-3.5 generated the identifier "Q186081.". Since no resource identifier for Wikidata contains punctuation and since removing punctuation is a task that can be easily implemented, we removed all punctuation from the returned resource identifiers before retrieving their labels.

Additionally, some of the generated resource identifiers such as the resource identifier for "HQ location" ("Q18674726") in the LC-QuAD 2.0 benchmark sample dataset's question "On which island is the HQ location of the Carlsberg Group?" do not exist on Wikidata.

The remaining resource identifiers generated range from being completely unrelated to the original phrase or term (e.g., "Q2074737" or "municipality of Spain" for "located on island", and "Q27020041" or "sports season" for "based in") to being partially correct (e.g.,"Q8072" or "volcano" for "volcanic eruption"). Even phrases for which Wikidata definitely has a corresponding resource identifier such as "Battle of Stalingrad" ("Q38789") will get linked to totally arbitrary resource identifiers by the model ("Q16709" which does not exist).

To obtain the labels of the resource identifiers generated for DBpedia we used the query shown in listing 38 where *cDBID* again represents the resource identifier whose label we want to retrieve.

```
SELECT ?label
WHERE {
    'cDBID' rdfs:label ?label.
```

```
    FILTER(LANG(?label)="en").
}
```

Listing 38: Query used to retrieve resource identifier labels from DBpedia.

Note that we did not check the DBpedia resource identifiers for added punctuation. This has not been done since resource identifiers for DBpedia already include punctuation, therefore limiting the possibilities of automatic removal of punctuation. However, left angle brackets and right angle brackets were added to the identifiers, since the GPT-model did not add them itself.

The generated identifiers for the two benchmark sample datasets show that GPT-3.5 performs even worse on DBpedia than on Wikidata.

Most of the generated resource identifiers returned no result at all when querying their label.

For the LC-QuAD 2.0 benchmark sample dataset, only the phrase "located on island" in the question "On which island is the HQ location of the Carlsberg Group" returned "dbo:island" which is the resource identifier for "island". For the QALD-9 benchmark sample dataset, on the other hand, none of the resource identifiers generated by GPT-3.5 returned a label.

In line with the results of the previous tasks, GPT-4 had fewer problems with conforming to our formatting instructions. However, GPT-4 did occasionally add a text in front of the resource identifier stating that it is the resource identifier for the given term or question.

While GPT-3.5 consistently made up resource identifiers, its successor often stated that the KG in question does not have a specific resource identifier for the given phrase or term. While this might seem like an improvement, a few problems remained.

First, Wikidata would still link wrong resource identifiers to certain phrases or terms. For example, for the term "Communist Party of China" GPT-4 generated the Wikidata resource identifier "Q7838" which represents the language Swahili. Furthermore, GPT-4 still made up resource identifiers that do not exist, such as "Q294466" for the term "chairperson". Last, some of the terms or phrases for which GPT-4 states that no resource identifier exists for the given KG do in fact have a corresponding resource identifier on said KG. This was for example the case with "Antonio José de Sucre". While the politician has in fact a specific resource identifier for Wikidata (Q189779) the LLM stated that this was not the case.

On the other hand, GPT-4 generated significantly more P-identifiers than its predecessor.

5.2.2 Global Phrase Mapping

When it comes to DBpedia, GPT-3.5 seemingly starts adding or changing numbers of its resource identifiers in order to forcefully generate exactly 10 results. Take a look at table 14 for instance, which shows the resource identifiers generated for the question "What is the foundational document of the Soviet Union?". While the first five identifiers represent real resources found on DBpedia, the following 5 identifiers are simply variations of the previous identifiers containing numbers.

Table 14: Resource identifiers generated by GPT-3.5 for the question "What is the foundational document of the Soviet Union?".

Resource identifier
http://dbpedia.org/resource/Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/Declaration_of_the_Creation_of_the_USSR
http://dbpedia.org/resource/1924_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1936_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1977_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1925_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1931_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1937_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1947_Constitution_of_the_Soviet_Union
1http://dbpedia.org/resource/1978_Constitution_of_the_Soviet_Union

Upon trying to query the generated identifiers' labels, none of them returned any result.

For Wikidata, GPT-3.5 again showed various strange behaviors.

First, the model sometimes added text before and after the identifier, such as enumerations. The resource identifiers for the question "What is the foundational document of the Soviet Union?" from the LC-QuAD 2.0 benchmark sample dataset were preceded by the letter A followed by the numbers 1 to 10, and a colon. Since the formulation of Wikidata's identifiers is very strict and simple, we isolated all Q- and P-identifiers from the LLMs' results before querying the resource identifiers' labels.

Similar to the direct phrase mapping results, GPT-3.5 would often use one of the generated identifiers, increase its number by one, and use the new identifier as a new result. Take for example the resource identifiers for the question "What is the foundational document of the Soviet Union?" as shown in table 15. While the first two resource identifiers differ significantly, the

following 8 identifiers are simply increments of the previous identifier, and therefore vary strongly in context.

Table 15: Resource identifiers and their label for the question "What is the foundational document of the Soviet Union?" generated by GPT-3.5.

Resource identifier	Label
NA	NA
A1: Q1321	Spanish
A2: Q1860	English
A3: Q1861	Bangkok
A4: Q1862	Wolin
A5: Q1863	Andorra le Vella
A6: Q1864	NA
A7: Q1865	Kuala Lumpur
A8: Q1866	Bangka Belitung Islands
A9: Q1867	Taipei
A10: Q1868	Paul Otlet

On the same note as adding enumerations, GPT-3.5 also sometimes did add labels to the generated resource identifiers. An example for this is shown in table 16 which shows the resource identifiers for the question "who is the record label and genre of The_Velvet_Underground?" and their labels.

Table 16: Resource identifiers and their label for the question "who is the record label and genre of The_Velvet_Underground?" generated by GPT-3.5.

Resource identifier	Label
Q7366 (The Velvet Underground)	song
P264 (record label)	record label
P136 (genre)	genre
Q334 (rock music)	Singapore
Q18953 (experimental rock)	Peter DeLuise
Q18963 (art rock)	Danish Touringcar Championship
Q18967 (proto-punk)	National Lampoon's Van Wilder
Q18968 (noise rock)	James DeGale
Q18969 (avant-garde)	Bicknell's Thrush
Q18970 (psychedelic rock)	Ambon City

While these labels give us a hint of what part of the question was used to construct each resource identifier, they also show that most of the generated

resource identifiers do not fit their corresponding phrase or term at all.

For some questions, such as the question "Which is the Wikimedia category for the category of associated people of Oslo?" in the LC-QuAD 2.0 benchmark sample dataset, GPT-3.5 just returned the same resource identifier ("Q1207682") 10 times.

For other questions however, the results are returned in some sort of logical form. Take for example the question "what is destroâ€™s gender?" from the SimpleQuestions benchmark sample dataset. For this question, GPT-3.5 returned "Q131336: P21 (gender) of Q370346" and "Q370346: P21 (gender) of Q131336". Both results were returned five times.

Also, for the question "who won the eurovision song contest 2016" in the WDAquaCore0Questions benchmark sample dataset, the resource identifier returned was followed by its Wikidata URL ("Q180684: https://www.wikidata.org/wiki/Q180684"). Again, this identifier has been returned 10 times.

Last, GPT-3.5 sometimes returned all identifiers in one line such as "Q1048, Q1048, Q1048, Q1048, Q1048, Q1048, Q1048, Q1048, Q1048, Q1048" for the LC-QuAD 2.0 benchmark sample dataset's question "On which island is the HQ location of the Carlsberg Group?", and often included empty results. Adjustments for this have been made in the code used to retrieve the generated resource identifiers' labels.

Compared to GPT-3.5, GPT-4 did not tend to just increase or change the numbers found in its resource identifiers. For the question "What is the foundational document of the Soviet Union?" from the LC-QuAD 2.0 benchmark sample dataset, for example, GPT-4 was able to generate a larger variety of resource identifiers related to the question. These identifiers are shown in table 17.

Table 17: Resource identifiers generated by GPT-4 for the question "What is the foundational document of the Soviet Union?".

Resource Identifier
http://dbpedia.org/resource/Soviet_Union
http://dbpedia.org/resource/Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1924_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1936_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/1977_Constitution_of_the_Soviet_Union
http://dbpedia.org/resource/Founding_of_the_Soviet_Union
http://dbpedia.org/resource/History_of_the_Soviet_Union
http://dbpedia.org/resource/Politics_of_the_Soviet_Union
http://dbpedia.org/resource/Government_of_the_Soviet_Union
1http://dbpedia.org/resource/Legal_history_of_the_Soviet_Union

For Wikidata, however, GPT-4 still sometimes failed to comply with the given formatting instructions. For the same question, GPT-4 added what it believes to be the identifiers name in front of it (e.g.,"Soviet Union - Q15180").

However, the resource identifiers generated by GPT-4 were as likely to be wrong as those generated by its predecessor, despite the model stating their label in front of them. An example for this can be found in the same question, where GPT-4 used the resource identifier "Q1237733" for the term "Declaration of the Creation of the USSR". Yet, this resource identifier represents the German artist "Hans Kloss".

Similar to the direct phrase mapping task, GPT-4 acknowledged if it believed that certain resource identifiers do not exist for the KG in question. An example for this would be the question "What is Loop ID for Simon Baron-Cohen?" from the LC-QuAD 2.0 benchmark sample dataset. For this question, GPT-4 stated that the term "Loop ID" does not have a corresponding resource identifier on Wikidata and instead returned other identifiers for Simon Baron-Cohen, including his supposed ORCID and his Wikidata identifier ("Q334029"). However, both were incorrect, with the latter instead representing the Battle of Bunker Hill.

The lack of correct resource identifier generated by both GPT-3.5 and GPT-4 using either single phrases or whole questions indicate that the LLMs lack knowledge about the underlying KGs and are not well suited to execute phrase mapping tasks in a KGQA-context.

5.3 Query Construction

Upon analyzing the results of the "Query Construction" phase, the differences between GPT-3.5 and GPT-4 became even more apparent. This section is therefore aimed at describing how the models differ wrt. constructing SPARQL queries to answer natural language questions.

5.3.1 Direct Construction

While GPT-3.5 was able to generate syntactically correct queries for almost all questions across all of the benchmark sample datasets, most of these queries did not return any result when executed. However, most of the returned results were correct answers to the generated queries. Table 18 shows the number of queries for each category across all benchmark sample datasets.

Table 18: GPT-3.5's results of direct query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	12	9	2
SimpleQuestions	14	11	3
QALD-9	10	8	2
WDAquaCore0Questions	15	8	6
LC-QuAD 2.0 (Wikidata)	15	14	0
LC-QuAD 2.0 (DBpedia)	14	11	2
Sum	80	61	13

One notable observation was that most of the correct queries generated by GPT-3.5 return the exact same answer as the benchmark's sample solution query. Take for example the question "Give me a list of all lakes in Denmark." from the QALD-9 benchmark for which the results of both the query generated by GPT-3.5, and the sample solution query return the same results in the identical order. While this might lead to the assumption that the LLM already knew the solution of the sample solution query due to the benchmark's popularity, a comparison of the two queries shown in listing 39 and listing 40 show that this was seemingly not the case. Hence, GPT-3.5 is in fact capable of generating proper SPARQL-queries for at least some of the most popular benchmarks' questions without the aid of additional information.

```
SELECT ?lake
WHERE {
  ?lake rdf:type dbo:Lake ;
```

```
        dbo:country dbr:Denmark .
}
```

Listing 39: GPT-3.5 generated query to answer the question "Give me a list of all lakes in Denmark.".

```
SELECT DISTINCT ?uri WHERE {
    {
        ?uri a <http://dbpedia.org/ontology/Lake> ;
            <http://dbpedia.org/ontology/country> <http://dbpedia.org/
                resource/Denmark>
    } UNION {
        ?uri a <http://dbpedia.org/class/yago/LakesOfDenmark>
    }
}
```

Listing 40: Sample solution query to answer the question "Give me a list of all lakes in Denmark.".

Looking at GPT-4, the results for this task show that while the number of syntactically correct queries generated by GPT-4 was far below that of GPT-3.5, the number of queries resulting in nothing also decreased significantly. By far the biggest improvement however can be seen in the number of queries yielding the correct result, going from 13 correct queries generated by GPT-3.5 up to 24 correct queries generated by GPT-4. A summary of results of GPT-4 in the context of direct query generation is shown in table 19.

Table 19: GPT-4's results of direct query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	10	5	3
SimpleQuestions	14	7	5
QALD-9	13	5	8
WDAquaCore0Questions	11	4	5
LC-QuAD 2.0 (Wikidata)	1	12	1
LC-QuAD 2.0 (DBpedia)	13	10	3
Sum	62	43	24

5.3.2 URI-Fed Construction

While one could assume that it would be easier for the LLMs to generate fitting SPARQL-queries if supplied with the correct resource identifiers, this was seemingly not the case. Despite the models' high success rate, wrt.

generating syntactically correct queries in the direct query construction task, the number of syntactically incorrect queries decreased slightly when the correct resource identifiers were supplied to the LLMs. However, the number of queries returning no result significantly decreased, indicating an increase in the models' accuracy when constructing SPARQL-queries. Most notably, however, most of the SPARQL-queries generated by GPT-3.5 that did return a result can be considered correct. Therefore, GPT-3.5 performs significantly better at generating fitting SPARQL-queries when being supplied with the correct resource identifiers. Table 20 again shows the number of queries for each category across all benchmark sample datasets.

Table 20: GPT-3.5's results of URI fed query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	14	4	10
SimpleQuestions	15	2	13
QALD-9	10	2	1
WDAquaCore0Questions	15	8	3
LC-QuAD 2.0 (Wikidata)	15	8	7
LC-QuAD 2.0 (DBpedia)	8	7	1
Sum	76	31	35

While the number of syntactically correct queries generated by GPT-3.5 decreased slightly when supplying the model with the correct resource identifiers, GPT-4's results improved in every category. Not only did the number of syntactically correct queries increase, but the number of empty queries almost halved when the correct resource identifiers were supplied. Last, also the number of queries returning the correct result increased drastically. A summary of GPT-4's results of the URI-fed query construction task is shown in table 21.

Table 21: GPT-4's results of URI fed query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	12	2	10
SimpleQuestions	14	1	13
QALD-9	13	4	9
WDAquaCore0Questions	15	5	7
LC-QuAD 2.0 (Wikidata)	15	6	6
LC-QuAD 2.0 (DBpedia)	9	7	2
Sum	78	25	47

5.3.3 Template-Based Construction

While supplying resource identifiers to the LLMs drastically improved the SPARQL-queries generated by GPT-3.5, the usage of SPARQL-templates had the opposite effect. While the number of syntactically correct or empty queries decreased only slightly, almost none of the queries returning a result returned the correct result. The constantly reoccurring problem was that GPT-3.5 was unable to replace the templates' slots by the correct resource identifiers. For the question "What is the foundational document of the Soviet Union?" in the LC-QuAD 2.0 benchmark sample dataset for example, the resulting query shown in listing 41 follows the SPARQL-template for a simple question, yet none of the slots have been replaced.

```
SELECT DISTINCT ?x WHERE {
  ?x ?p ?y .
  ?x rdf:type ?c .
}
```

Listing 41: Template-based SPARQL-query generated by GPT-3.5 to answer the question "What is the foundational document of the Soviet Union?".

One could therefore imagine that the LLM might need to be supplied with additional instructions of how to fill in the supplied SPARQL-template in order to generate more fitting queries.

A summary of the results of the template-based query construction task using GPT-3.5 is shown in table 22.

Table 22: GPT-3.5's results of template-based query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	11	7	0
SimpleQuestions	14	7	0
QALD-9	10	2	1
WDAquaCore0Questions	14	1	1
LC-QuAD 2.0 (Wikidata)	11	7	0
LC-QuAD 2.0 (DBpedia)	10	2	1
Sum	70	26	3

The results for GPT-4 were in line with the results of the previous sections. Again, the number of correct queries increased drastically, while the number of queries not returning a result decreased. A summary of GPT-4's results for the task of template-based query construction is shown in table 23.

Table 23: GPT-4's results of template-based query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	9	8	1
SimpleQuestions	15	11	2
QALD-9	13	5	5
WDAquaCore0Questions	14	2	8
LC-QuAD 2.0 (Wikidata)	14	12	1
LC-QuAD 2.0 (DBpedia)	14	10	3
Sum	79	48	20

However, upon inspecting the generated queries, it is unclear whether GPT-4 does in fact make use of the provided SPARQL-template or if it generates the queries itself. Again, take a look at the query generated for the question "What is the foundational document of the Soviet Union?" from the LC-QuAD 2.0 for Wikidata shown in listing 42.

```
SELECT DISTINCT ?document ?documentLabel WHERE {
  wd:Q15180 wdt:P457 ?document.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_
      LANGUAGE],en". }
}
```

Listing 42: Template-based SPARQL-query generated by GPT-4 to answer the question "What is the foundational document of the Soviet Union?".

While returning a result that can be considered correct, the query does not hold much similarity to the SPARQL-template provided.

5.3.4 URI-Fed Template-Based Construction

As expected, supplying the correct resource identifiers to the LLM did not improve its performance, since the problem of it being unable to replace the slots in the SPARQL-templates still remained. Supplying the correct resource identifiers to GPT-3.5 even worsened its performance, leading to even less executable queries and only one query returning the correct answer. Table 24 shows GPT-3.5's results of the URI-fed template-based query construction task.

Table 24: GPT-3.5's results of URI-fed template-based query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	10	8	1
SimpleQuestions	10	9	0
QALD-9	10	7	1
WDAquaCore0Questions	10	10	0
LC-QuAD 2.0 (Wikidata)	12	12	0
LC-QuAD 2.0 (DBpedia)	11	10	0
Sum	63	56	1

Contrary to its predecessor, GPT-4's performance when using SPARQL-templates did increase significantly once the model was supplied with the correct resource identifiers. With the number of queries returning correct answers being more than twice the number of correct queries generated by GPT-3.5, it is clear that the problems GPT-3.5 faced when being instructed to use a SPARQL-template do not hinder GPT-4 similarly. However, with the number of syntactically correct, empty and correct queries being close for both the URI-fed construction, and the URI-fed template based construction task, the suspicion that GPT-4 might simply ignore the provided templates, raised in the previous section, cannot be neglected. The summary of GPT-4's results for the URI-fed template-based query construction task is shown in table 25.

Table 25: GPT-4's results of URI-fed template-based query construction by category.

Benchmark	Executable	Empty	Correct
StudentQuestions	14	1	13
SimpleQuestions	12	1	11
QALD-9	13	5	8
WDAquaCore0Questions	13	4	3
LC-QuAD 2.0 (Wikidata)	13	5	8
LC-QuAD 2.0 (DBpedia)	12	11	1
Sum	77	27	44

5.4 Technical Performance

Besides issues such as the GPT-models not conforming to the given instructions, other problems wrt. reliability occurred frequently during this study. Most of these problems came in the form of errors, such as the error for

failed request to the OpenAI API (Error 503) or the error for when the host *api.openai.com* could not be resolved.

Additionally, GPT-4 often did not respond to its requests. Yet, the LLM did not return any error messages in these cases. While this behavior was mostly limited to the questions in the WDAquaCore0Questions benchmark sample dataset, it strongly impacts the LLM's reliability when used in loops or being nested in larger problems, since these would not stop their operations.

Aside from the evaluations of the models' generated content and their stability, we also recorded the execution time for each task. This means that for each sample benchmark dataset, the time it took each LLM to execute the given task across all questions was recorded. The different models' average execution time per question for each task is shown in table 26.

Table 26: Average execution time per question for each task in the KGQA-process (in seconds).

Task	GPT-3.5	GPT-4
Phrase variation	19.449	30.202
Question variation	1.643	5.331
Question pseudonymization	0.652	1.768
Type detection	0.487	1.036
POS tagging	1.86	4.762
Named entity recognition	0.757	1.682
Entity and relationship detection	1.071	2.321
Semantic parsing	3.679	3.963
Direct phrase mapping	2.123	4.402
Global phrase mapping	3.63	11.097
Direct construction	2.97	6.949
URI-fed construction	2.563	5.656
Template-based construction	2.564	5.956
URI-fed template-based construction	2.564	5.682
Average	3.287	6.486

Looking at these results, one can see that despite its lackluster stability, GPT-4 executed the different task at almost twice the speed of its predecessor.

While the task of phrase variation required a seemingly disproportional time to be executed, especially when compared to the question variation task, it must be noted that while each question in the question variation task

received only three variations in total, each question in the phrase variation task received three variations for each phrase within the question. Therefore, the task of phrase variation naturally requires more time to be executed than the task of varying the question, for example.

6 Limitations

The results of this study show that OpenAI's GPT-3.5 and GP-4 based models perform drastically differently wrt. KGQA-sub-tasks.

When it comes to simple natural language processing tasks in the KGQA-process such as phrase variation, type detection or POS tagging, both GPT-models performed rather well. This might be to be expected due to the fact that these models have been developed to succeed on exactly such tasks. However, we were able to observe some inconsistencies wrt. some of these tasks, such as using different tag sets in the POS tagging task. We do believe, however, that these artifacts can be controlled if sufficient information such as which tag set to use and a detailed instruction on how to execute the given task would be supplied to the model via the prompt. Regardless, GPT-3.5's arguably biggest weakness lied in its inconsistency and instability. Failing to conform to the given formatting instructions drastically decreases the model's potential value for being implemented in real-life applications, such as publicly available KGQA-systems. While the problem still occurred when using GPT-4, it did so at a much smaller scale, making the newer model much more reliable in this regard. Here it must be noted that OpenAI now allows users to fine-tune GPT-3.5 by enabling developers to customize the model to their needs, achieving improved reliability and more reliable output formatting, hence directly counteracting the observed issues [4]. At the time of writing, the ability to fine-tune the GPT-4 model has been announced, jet not enabled.

On the other hand, both GPT-3.5 and GPT-4 were unable to reliably link keywords or phrases to their corresponding resource identifiers in the underlying KGs. This could be explained by the models simply lacking sufficient knowledge about most resource identifiers in the chosen KGs (i.e., resource identifiers that have not been commonly used in example queries). With most resource identifiers being either unrelated to the original term or phrase or non-existent, it is safe to say that neither GPT-3.5 nor GPT-4 are suited for Phrase Mapping tasks, as described by Diefenbach et al. [17].

Very different results were observed during the effort of using the GPT-models to construct SPARQL-queries. Here, GPT-3.5 did not perform well when constructing SPARQL-queries without additional information besides

the question to be answered. Once supplied with the correct resource identifiers needed to answer the question, the model's results improved significantly. However, these results were still far from astonishing. Supplying SPARQL-templates to the model even worsened the quality of the generated queries, even once a SPARQL-template and the necessary resource identifiers were given to the LLM. From the generated queries, it can be assumed that GPT-3.5 was not able to fill out the given templates. These results indicate that using GPT-3.5 should not necessarily be considered over other methods to generate SPARQL-queries. GPT-4 on the other hand outperformed GPT-3.5 in every approach taken to generate SPARQL-queries using it. However, since the LLM delivered similar results when using SPARQL-templates and when not and the large difference between the given templates and the generated queries, we have reason to believe that GPT-4 might not have used the given templates at all when generating queries. Despite this, GPT-4 performed well enough for us to believe that the LLM has potential to be used in the query construction task within the KGQA-process.

Besides the LLMs' struggles to conform to instructions mentioned before, various problems wrt. the API's stability strongly limit the models' potential use in applications requiring high degrees of reliability, such as publicly available KGQA-systems.

Last, while certain questions such as many of those found in the WDAqua-Core0Questions benchmark propose problems for KGQA-systems as a whole, we did not observe any patterns indicating that certain question types are better suited to evaluate LLM-based KGQA-systems over others.

Despite these findings seemingly strong indication of a significant superiority of GPT-4 over its predecessor and for which tasks in the KGQA-process these LLMs could be reasonably used, they must be considered with care. As Chen et al. noted, one should closely monitor the behavior of these LLMs as their performance wrt. various tasks can change substantially, even over a short time frame [12].

Nonetheless, the results of this study add a valuable foundation to the still scarce body of literature related to the usage of LLMs in KGQA.

7 Conclusion and Further Research

Evaluating both GPT-3.5 and GPT-4 over multiple tasks commonly found in KGQA-systems showed that both GPT-3.5 and GPT-4 yielded promising

results, wrt. NLP tasks. When it comes to linking keywords and phrases to the resource identifiers of an underlying KG, neither of the two models showed any sign of being suited for this task. While GPT-3.5 struggled with the creation of SPARQL-queries to answer natural language questions, GPT-4 showed a large improvement and could be considered a viable option for this task. However, both models struggled with issues related to reliability in the form of following the given instructions and the stability of OpenAI's API.

Evaluation of the models on different samples of popular benchmark datasets showed that the LLMs are not particularly challenged by any specific question type, but rather by poorly formulated questions and ones that hardly resemble a question at all.

While a KGQA-system consisting completely of GPT-3.5 or GPT-4 instances seems unreasonable following these results, LLMs can still be considered viable options for certain tasks in the KGQA-process, leaving room for further studies in this field. Future work should therefore consider the following aspects.

First, due to the varying quality of benchmark questions a set of less ambiguous questions that leave little to no room for misinterpretation should be used when employing GPT models in the KGQA context. This point becomes especially relevant if no context narrowing down the scope of possible interpretations and therefore possible answers is being provided to the model.

Second, researchers should take into account the instability of the GPT models when conducting their work. Although it can be expected that popular models such as GPT-3.5 and GPT-4 will become more stable and reliable as time goes on, other alternatives able to counteract the problems we encountered in this study could consist of using locally ran LLMs, such as LLaMa [56].

Given the general and differentiated nature of this experiment, researchers are advised to fine tune their prompts when using LLMs such as the GPT models in any KGQA tasks. Specifically when using templates to construct SPARQL queries, additional instructions on how to use the given templates could drastically increase the change of LLMs construction functional SPARQL queries, leaving prompt tuning specific to KGQA to be a promising topic for further research.

Last, the GPT models have been evaluated fundamentally and in isolation. While the models did not perform well in all components of the KGQA-process, the results have been promising for certain tasks, such as the construction of SPARQL queries. Further research should therefore take the

models out of isolation and combine them with established KGQA-systems. This would allow for a more realistic assessment of whether the use of LLMs could improve the quality of KGQA-systems.

References

[1] DBpedia Linked Data Fragments. `https://fragments.dbpedia.org/`, 2023. Accessed: 2023-07-21.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report, 2023.

[3] Bobby Allyn. 'New York Times' considers legal action against openai as copyright tensions swir. `https://www.npr.org/2023/08/16/1194202562/new-york-times-considers-legal-action-against-openai-as-copyright-tensions-swirl`, 2023. Accessed: 2023-08-17.

[4] John Allard Logan Kilpatrick Andrew Peng, Michael Wu and Steven Heidel. Gpt-3.5 turbo fine-tuning and api updates. `https://openai.com/blog/gpt-3-5-turbo-fine-tuning-and-api-updates`. Accessed: 2023.08.

[5] Jonathan Berant, Andrew K. Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Conference on Empirical Methods in Natural Language Processing*, 2013.

[6] Bettina Berendt, Andreas Hotho, Dunja Mladenic, Maarten van Someren, Myra Spiliopoulou, and Gerd Stumme. A roadmap for web mining: From web to semantic web. In *Web Mining: From Web to Semantic Web*, pages 1–22. Springer Berlin Heidelberg, 2004.

[7] Margaret A Boden. *Artificial intelligence*. Elsevier, 1996.

[8] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks, 2015.

[9] 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, 2020.

[10] Jorge Cardoso. The semantic web vision: Where are we? *IEEE Intelligent Systems*, 22(5):84–88, September 2007.

[11] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models, 2023.

[12] Lingjiao Chen, Matei Zaharia, and James Zou. How is chatgpt's behavior changing over time?, 2023.

[13] Hai Cui, Tao Peng, Lizhou Feng, Tie Bao, and Lu Liu. Simple question answering over knowledge graph enhanced by question pattern classification. *Knowledge and Information Systems*, 63(10):2741–2761, September 2021.

[14] D063520. Github - wdaquacore0questions. `https://github.com/WDAqua/WDAquaCore0Questions`, 2017. Accessed: 2023-02-28.

[15] Aarthi Dhandapani and Viswanathan Vadivel. Question answering system over semantic web. *IEEE Access*, 9:46900–46910, 2021.

[16] Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. Towards a question answering system over the semantic web, 2018.

[17] Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems*, 55(3):529–569, September 2017.

[18] Dennis Diefenbach, Kamal Singh, and Pierre Maret. WDAqua-core0: A question answering component for the research community. In *Semantic Web Challenges*, pages 84–89. Springer International Publishing, 2017.

[19] Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret. Question answering benchmarks for wikidata. In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017.*, 2017.

[20] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. LC-QuAD 2.0: A large dataset for complex question answering over wikidata and DBpedia. In *Lecture Notes in Computer Science*, pages 69–78. Springer International Publishing, 2019.

[21] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing wikidata to the linked data web. In *The Semantic Web – ISWC 2014*, pages 50–65. Springer International Publishing, 2014.

[22] W. Floyd, T. Kleber, M. Pasli, J.J. Qazi, C.C. Huang, J.X. Leng, B. Ackerson, D.J. Carpenter, J.K. Salama, and M.J. Boyer. Evaluating the reliability of chat-gpt model responses for radiation oncology patient inquiries. *International Journal of Radiation Oncology*Biology*Physics*, 117(2):e383, October 2023.

[23] Samanyou Garg. ChatGPT alternatives that will blow your mind in 2023 — writesonic.com. https://writesonic.com/blog/chatgpt-alternatives/, 2023. Accessed: 2023-05-06.

[24] Jorão Gomes, Rômulo Chrispim de Mello, Victor Ströele, and Jairo Francisco de Souza. A study of approaches to answering complex questions over knowledge bases. *Knowledge and Information Systems*, 64(11):2849–2881, August 2022.

[25] Alex Graves. Sequence transduction with recurrent neural networks, 2012.

[26] Eleni Ilkou and Maria Koutraki. Symbolic vs sub-symbolic ai methods: Friends or enemies? 11 2020.

[27] Longquan Jiang and Ricardo Usbeck. Knowledge graph question answering datasets and their generalizability. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, July 2022.

[28] Weiqiang Jin, Biao Zhao, Hang Yu, Xi Tao, Ruiping Yin, and Guizhong Liu. Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning. *Data Mining and Knowledge Discovery*, 37(1):255–288, November 2022.

[29] Douglas Johnson, Rachel Goodman, J Patrinely, Cosby Stone, Eli Zimmerman, Rebecca Donald, Sam Chang, Sean Berkowitz, Avni

Finn, Eiman Jahangir, Elizabeth Scoville, Tyler Reese, Debra Friedman, Julie Bastarache, Yuri van der Heijden, Jordan Wright, Nicholas Carter, Matthew Alexander, Jennifer Choe, Cody Chastain, John Zic, Sara Horst, Isik Turker, Rajiv Agarwal, Evan Osmundson, Kamran Idrees, Colleen Kiernan, Chandrasekhar Padmanabhan, Christina Bailey, Cameron Schlegel, Lola Chambless, Mike Gibson, Travis Osterman, and Lee Wheless. Assessing the accuracy and reliability of ai-generated medical responses: An evaluation of the chat-gpt model. February 2023.

[30] Haemin Jung and Wooju Kim. Automated conversion from natural language query to SPARQL query. *Journal of Intelligent Information Systems*, 55(3):501–520, January 2020.

[31] Gerhard Klager and Axel Polleres. Is GPT fit for kgqa? - preliminary results. In Sanju Tiwari, Nandana Mihindukulasooriya, Francesco Osborne, Dimitris Kontokostas, Jennifer D'Souza, Mayank Kejriwal, and Edgard Marx, editors, *Joint Proceedings of the Second International Workshop on Knowledge Graph Generation From Text and the First International BiKE Challenge co-located with 20th Extended Semantic Conference (ESWC 2023), Hersonissos, Greece, May 29th, 2023*, volume 3447 of *CEUR Workshop Proceedings*, pages 171–191. CEUR-WS.org, 2023.

[32] Ben Krose. *An introduction to neural networks*. 1996.

[33] Shiqi Liang, Kurt Stockinger, Tarcisio Mendes de Farias, Maria Anisimova, and Manuel Gil. Querying knowledge graphs in natural language. *Journal of Big Data*, 8(1), January 2021.

[34] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Web Semantics Science Services And Agents On The World Wide Web*, 21:3–13, 2013.

[35] Angel R. Martinez. Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(1):107–113, September 2011.

[36] Conan Mercer. The rise of chat gpt: The future of conversational ai. `https://medium.com/@conan.mercer/the-rise-of-chat-gpt-the-future-of-conversational-ai-91622b9db303`, 2023. Accessed: 2023-05-06.

[37] Merriam-Webster. Slugger definition meaning. `https://www.merriam-webster.com/dictionary/slugger`, 2023. Accessed: 2023-08-10.

[38] Salman Mohammed, Peng Shi, and Jimmy Lin. Strong baselines for simple question answering over knowledge graphs with and without neural networks, 2017.

[39] Behrang Mohit. Named entity recognition. In *Natural Language Processing of Semitic Languages*, pages 221–245. Springer Berlin Heidelberg, 2014.

[40] Wardani Muhamad, Suhardi, and Yoanes Bandung. Transforming OpenAPI specification 3.0 documents into RDF-based semantic web services. *Journal of Big Data*, 9(1), April 2022.

[41] Jere Odell, Mairelys Lemus-Rojas, and Lucille Brys. Wikidata data model, 2022.

[42] Inc. Ontotext USA. What is a knowledge graph? `https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/`, 2023. Accessed: 2023-09-07.

[43] Abdelghny Orogat, Isabelle Liu, and Ahmed El-Roby. Cbench: Towards better evaluation of question answering over knowledge graphs, 2021.

[44] Seonyeong Park, Hyosup Shim, and Gary Geunbae Lee. Isoft at qald-4: Semantic similarity-based question answering system over linked data. In *Conference and Labs of the Evaluation Forum*, 2014.

[45] Michael Petrochuk and Luke Zettlemoyer. SimpleQuestions nearly solved: A new upperbound and baseline approach. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

[46] Mikhail Popov. *WikidataQueryServiceR: API Client Library for 'Wikidata Query Service'*, 2020. R package version 1.0.0.

[47] Contributors to Wikimedia projects. Wikibase/datamodel - mediawiki, Dec 2023.

[48] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF — w3.org. `https://www.w3.org/TR/rdf-sparql-query/`, 2008. Accessed: 2023-09-07.

[49] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022.

[50] Aleksandr Perevalov Longquan Jiang Julius Schulz Angelie Kraft Cedric Moeller Junbo Huang Jan Reineke Axel-Cyrille Ngonga Ngomo Muhammad Saleem Andreas Both Ricardo Usbeck, Xi Yan. Qald-10 - the 10th challenge on question answering over linked data. Under submission., 2023.

[51] Brian Richards. Type/token ratios: what do they really tell us? *Journal of child language*, 14:201–9, 07 1987.

[52] Iegor Rudnytskyi. *openai: R Wrapper for OpenAI API*, 2023. R package version 0.4.0.

[53] Uma Sawant, Saurabh Garg, Soumen Chakrabarti, and Ganesh Ramakrishnan. Neural architecture for question answering using a knowledge graph and web corpus. *Information Retrieval Journal*, 22(3-4):324–349, January 2019.

[54] Sangjin Shin, Xiongnan Jin, Jooik Jung, and Kyong-Ho Lee. Predicate constraints based question answering over knowledge graph. *Information Processing &amp Management*, 56(3):445–462, May 2019.

[55] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: An overview. In *Treebanks*, pages 5–22. Springer Netherlands, 2003.

[56] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[57] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. 10 2017.

[58] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Sparql template-based question answering. 01 2012.

[59] Ricardo Usbeck, Ria Gusmita, Muhammad Saleem, and Axel-Cyrille Ngonga Ngomo. 9th challenge on question answering over linked data (qald-9). 11 2018.

[60] Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. Message passing for complex question answering over knowledge graphs. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, November 2019.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[62] Andra Waagmeester, Gregory Stupp, Sebastian Burgstaller-Muehlbacher, Benjamin M Good, Malachi Griffith, Obi L Griffith, Kristina Hanspers, Henning Hermjakob, Toby S Hudson, Kevin Hybiske, Sarah M Keating, Magnus Manske, Michael Mayers, Daniel Mietchen, Elvira Mitraka, Alexander R Pico, Timothy Putman, Anders Riutta, Nuria Queralt-Rosinach, Lynn M Schriml, Thomas Shafee, Denise Slenter, Ralf Stephan, Katherine Thornton, Ginger Tsueng, Roger Tu, Sabah Ul-Hasan, Egon Willighagen, Chunlei Wu, and Andrew I Su. Science forum: Wikidata as a knowledge graph for the life sciences. *eLife*, 9:e52614, mar 2020.

[63] Ruijie Wang, Meng Wang, Jun Liu, Michael Cochez, and Stefan Decker. Structured query construction via knowledge graph embedding. *Knowledge and Information Systems*, 62(5):1819–1846, September 2019.

[64] Wikipedia. Atoll — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Atoll&oldid=1165307990`, 2023. [Accessed: 2023-08-10].

[65] Wikipedia. Tagesschau (German TV programme) — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Tagesschau%20(German%20TV%20programme)&oldid=1161090999`, 2023. Accessed: 2023-07-19.

[66] Mohammad Yani and Adila Alfa Krisnadhi. Challenges, techniques, and trends of simple knowledge graph question answering: A survey. *Information*, 12(7):271, June 2021.

[67] Mohammad Yani, Adila Alfa Krisnadhi, and Indra Budi. A better entity detection of question for knowledge graph question answering through extracting position-based patterns. *Journal of Big Data*, 9(1), June 2022.

[68] Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models, 2023.

[69] Munazza Zaib, Wei Emma Zhang, Quan Z. Sheng, Adnan Mahmood, and Yang Zhang. Conversational question answering: a survey. *Knowledge and Information Systems*, 64(12):3151–3195, September 2022.

A Is GPT fit for KGQA? â€“ Preliminary Results

Is GPT fit for KGQA? – Preliminary Results*

Gerhard G. Klager[1,*,†], Axel Polleres[1,†]

[1]WU Wien - Vienna University of Economics and Business, Welthandelsplatz 1, Vienna, 1020, Austria

Abstract

In this paper we report about preliminary results on running question answering benchmarks against the recently hyped conversational AI services such as ChatGPT: we focus on questions that are known to be possible to be answered by information in existing Knowledge graphs such as Wikidata. In a preliminary study we experiment, on the one hand, with questions from established KGQA benchmarks, and on the other hand, present a set of questions established in a student experiment, which should be particularly hard for Large Language Models (LLMs) to answer, mainly focusing on questions on recent events. In a second experiment, we assess how far GPT could be used for query generation in SPARQL. While our results are mostly negative for now, we hope to provide insights for further research in this direction, in terms of isolating and discussing the most obvious challenges and gaps, and to provide a research roadmap for a more extensive study planned as a current master thesis project.

Keywords

Question Answering, KGQA, LLMs, GPT

1. Introduction

With the ever-growing number of publicly available Knowledge Graphs and their increasing relevancy the task of question answering (KGQA from here on) has risen in popularity as well [1]. The purpose of a KGQA-system is to allow end-users to retrieve information stored in a KG by means of natural language questions, without being familiar with the KG's structure or the query language used to access said KG.

In order to achieve this goal, often some kind of translation of natural language questions into a query is taking place [2]. Many different KGQA approaches exist, ranging from template-based approaches [3] to approaches based on unsupervised message passing [4] or approaches using methods of machine learning [5]. The capabilities of KGQA-systems range from answering simple questions [6] to complex questions [7] as well to engage in single- and multi-turn (or conversational) question answering [8]. Additionally, some of these approaches even try to enable QA independent of a fixed KG or language [2].

To train and evaluate these models numerous benchmarks have been created, enabling a direct

TEXT2KG 2023: Second International Workshop on Knowledge Graph Generation from Text, May 28 - Jun 1, 2023, co-located with Extended Semantic Web Conference (ESWC), Hersonissos, Greece

*The text of this paper was hand-written without the support of text-generating AI, this – however – does not apply to the SPARQL query examples in this paper ;-)

*Corresponding author.

†These authors contributed equally.

gerhard.klager@gmail.com (G. G. Klager); axel.polleres@wu.ac.at (A. Polleres)

0009-0000-2816-219X (G. G. Klager); 0000-0001-5670-1146 (A. Polleres)

comparison between existing and new QA-systems [1].

At the same time, with the recent success of OpenAI's ChatGPT[9] and its many competitors [10], we see many applications of such large language models (LLMs), not only restricted to question answering alone, but also in producing more or less useful code in programming and query languages. Facing these developments, we may ask ourselves both (a) if such LLMs can act as serious contenders to bespoke KGQA systems, and (b) whether LLMS could be used as a supportive technology for query formulation in the context of KGQA. However, literature covering this subject is still scarce and end-to-end QA-systems using LLMs such as ChatGPT in a synergetic combination with KGQA have not yet been proposed in abundance.

The aim of this paper is therefore to fill this gap by exploring the possibilities of using LLMs such as ChatGPT in the task of KGQA and to challenge the status quo of existing benchmarks aimed at training and evaluating KGQA-systems.

In particular, we are interested in answering the following questions:

1. How does LLM-based QA differ from established KGQA approaches and what are the respective strengths, weaknesses and challenges of the two methods?
2. Which components used in KGQA-systems could be enhanced using LLMs?
3. What types of questions are found in existing benchmarks for KGQA approaches and in how far can these be used in benchmarking LLM-based QA approaches?
4. How can a comprehensive benchmark for LLM-based, KGQA-based but also combined QA approaches be constructed that is challenging the current weaknesses of both approaches?

In order to get closer to answering these questions, we have started with a comprehensive literature review. The goal of this literature review is to establish an overview of the already existing different QA-systems and benchmarks on the one hand and to lay the foundations of the approaches chosen to create our proposed new benchmark and QA-system as well as their characteristics and main components. Our initial collection of articles[2, 4, 11, 1, 8, 12, 13, 7, 6, 14, 15, 5, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41] contains both proposed solutions to
(KG)QA as well as existing benchmarks.

While still ongoing, we provide an overview of our current status of identified benchmarks and their main characteristics, as well as (components of) KGQA systems in Section 2 below. In the next step, we plan to compare existing benchmarks and systems from the literature for KGQA with LLM based question-answering. This shall be done by systematically comparing the questions available in each benchmark, wrt. the (syntactic) structures of the questions and – if available – corresponding structured queries, topic domains, and other characteristics indicating the complexity of the question answering tasks (such as for instance aggregations needed, etc.) and comparing the results of established KGQA and LLM based QA on each of these benchmarks.

For this first preliminary study, we proceed by selecting a mini-benchmark from

- sample subsets from the SimpleQuestions [42] and QALD-5 [43] benchmark datasets,
- a small dataset we manually designed to challenge ChatGPT

both of which having in mind that – in principle – the respective answers should intuitively be findable in existing KGs such as Wikidata [44]. We present this mini-benchmark in Section 3.

While certainly not yet representative of the field, we aim at drawing some initial conclusions about LLMs/GPTs capabilities and challenges with respect to two separate subtasks of KGQA

- directly answering the natural language queries from our mini-benchmark vs.
- translating the natural language queries to SPARQL

We summarize the performance on both these subtasks in Section 3.4 and derive hypothesis for further investigation.

We conclude in Section 4 with an outlook for further work that also include a workplan and main tasks to be executed in an ongoing Master thesis, wherefore we eagerly look forward to feedback during the TEXT2KG workshop. As an overall goal in our research agenda, we plan to design and implement a new hybrid approach making use of LLMs, such as ChatGPT (or also other emerging, and hopefully open LLMs) to improve upon the weaknesses of existing KGQA-systems without "LLM support".
Additionally, we hope our findings will serve as a base for new QA benchmarks aimed at improving the training and evaluation of future, combined LLM-and-KGQA-systems.

2. Related work on KGQA

With the growing attention given to KGQA in recent years we can also observe a large growth of literature covering KGQA-systems and in terms of different methods and benchmarks to evaluate these systems. This section is dedicated to provide an overview of this literature and laying the foundation for our planned research.

2.1. KGQA Benchmarks

The topic of benchmarking in QA ranges from the methods of creating benchmarking datasets, to the types of questions and queries used in a dataset, to methods to evaluate those benchmarks themselves.

Probably the most widely used family of question answering datasets is represented by the Question Answering over Linked Data (*QALD* from here on) campaign. This series of challenges aims at providing benchmarks for all QA-systems designed at using natural language requests of a user to retrieve information stored as structured data such as the RDF data format. Additionally, the challenge aims at comparing current state-of-the-art QA-systems with regard to their individual strengths and shortcomings. In order to participate in the current QALD challenge users can simply run their QA-system using the current challenge's dataset before storing their results in an XML file and upload it to the challenge's website [45]. The QALD challenge is currently taking place in its 10^{th} iteration [39] and accordingly provides 10 datasets that could be used within further elaborations of our study. A detailed analysis of these datasets is on our agenda. For the moment, we have considered a sample from the 5^{th} QALD [43] in our preliminary experiments, see below.

As a more manageable starting point, *SimpleQuestions* is a dataset containing 100k questions aimed at training and evaluating QA-systems with regard to solving the simple question answering problem, which consists answering questions that can be rephrased as (single triple) queries that ask for all objects linked to a questions given subject by its given relationship. In this context, simple QA is a term used referring to the simplicity of the reasoning process necessary to answer questions [46]. While SimpleQuestions was originally designed to be run over FreebBase, Diefenbach et al. [42] have adapted/extended the original SimpleQuestions dataset to Wikidata recently, which we include in our preliminary study since it is possible to be tested against a large KG, available via a public SPARQL endpoint.

As for further relevant QA benchmarks, Berant et. al. [47] created a new QA dataset named *WebQuestions*. The WebQuestions dataset acts as an extension to the FREE917 dataset (again based on Freebase) aimed at evaluating QA-systems. The authors created this dataset due to the FREE917 dataset requiring logical forms, making it inherently more difficult to scale it up due to the requirement of having expertise in annotating logical forms. Using the Google Suggest API, the authors obtained questions beginning with a wh-word (where, who, when, etc.) and containing exactly one entity. For each question, five candidate queries have been created. After collecting 1M questions in this process, 100k randomly selected questions have been submitted to Amazon Mechanical Turk (AMT from here on) where workers answered questions detecting duplicates and filtering out questions that could not be answered. The remaining dataset contained 5.810 questions. In particular, the approach to crowd-source and to cross-check labeling in order to compare humans against QA systems may also be useful for us in further elaborations of our study.

As an alternative to simple questions the Large-Scale Complex Question Answering Dataset 2.0 [40] (*LC-QuAD 2.0* from here on) is an extension to the original LCQuAD dataset [41] containing 30k complex questions as well as their corresponding paraphrased versions and SPARQL queries. This dataset is both compatible with Wikidata and DBpedia (2018). The dataset was created by generating a number of SPARQL queries before verbalizing them into natural language questions using the AMT. Afterward, these questions have been paraphrased to create additional natural language questions. The LC-Quad 2.0 dataset contains 10 different question types ranging from single fact questions which will be answered by returning either a subject or object to complex questions requiring complex patterns, temporal information to be answered, etc.

Another approach of how to create a benchmark has been taken by the creators of the WDAquaCore0Questions dataset which represents a collection of questions asked by users testing the demo of the WDAqua-core-0 QA-system for Wikidata [48].

Jiang and Usbeck analyzed 25 KGQA datasets with regard to five different KGs. Their study showed that many available KGQA datasets are unfit to train KGQA-systems due to their underlying assumptions or that these datasets are outdated and based on discontinued KGs. Additionally, the authors share light on the difficulties and high costs related to the generation of new datasets. Therefore, they propose an automated method to re-split datasets enabling their generalization as well as a method to analyze existing KGQA datasets with regard to their generalizability [11].

While many different benchmarks aimed at evaluating QA-systems for different KGs exist the question of which benchmark one should use can be a difficult one to answer. To answer this

question Orogat, Liu, and El-Roby proposed *CBench* [1], a suite that enables users to analyze existing benchmarks with regard to linguistic, syntactic, and structural properties of the dataset's questions and queries as well as to evaluate QA-systems. Additionally, the authors provide an overview of different creation methods for benchmarks ranging from manual creation based on heuristics to benchmarks created automatically from the KG in question.

In light of recent developments, and while social media is full of examples, there is — to the best of our knowledge — not yet a dedicated QA dataset originally tailored to LLMs and GPT specifically. In order to fill this gap, we asked students of the Digital Economy masters' program at the Vienna University of Economics and Business to generate a set of natural language questions aimed at asking ChatGPT to formulate queries GPT-3 would fail upon but suspected to be possible to answer with the information in publicly available KGs such as Wikidata. As a hint, we emphasized that we suspect LLMs to struggle with (a) recent events information beyond the training phase of the LLM (b) complex questions that require non-obvious conceptual understanding and reasoning. The students' task was to also find/formulate the corresponding SPARQL queries and – in the light of recent advances of LLMs for also code and query generation, attempt whether ChatGPT was able to create such queries. We report on a selected subset of these questions in section 3 below.

2.2. Question Answering Systems

With the existence of numerous QA-benchmarks it is no surprise that the literature presents an abundance of different QA-systems as well. These systems range from ones limited to single KGs to systems able to access multiple KGs, from language dependent to language independent systems, and from simpler template-based systems to complex systems incorporating elements of machine learning.

Diefenbach et. al. propose a QA-system capable of querying multiple KGs independent of the natural language used. Their approach has been evaluated on five well-known KGs and five different languages using three different benchmarks. Their proposed QA-system first performs *entity recognition* in terms of searching corresponding IRIs whose lexicalization is an n-gram (consecutive elements in a text) in the asked natural language questions question. After removing stop words from the set of IRIs queries that could represent possible interpretations of the question are constructed before being ranked based on multiple aspects such as the number of words matching the words in the original question. Next, a logistic regression based on labeled SPARQL queries will be trained to compute the confidence score for each query. Last, the highest ranked query above a certain threshold will be used to answer the question. If no query with confidence above the threshold is found, the whole question will be deemed unanswerable. During their study, the authors discovered performance differences in their approach wrt. different (natural) languages used and link these differences to the quality of the available data for each language [2].

Vakulenko et al. [4] take a quite different approach based on the usage of unsupervised message passing (QAmp from here on) which consists of two phases: in the first phase called question interpretation, the relevant sets of entities and predicates necessary for answering the

input question are again being identified and their confidence scores are being computed. In the second phase, the so-called answer inference phase, these confidence scores are propagated and aggregated over the underlying KG's structure, providing a confidence distribution over a set of possible answers which is then be used to locate the corresponding answer entities, rather than translating the query to SPARQL.

Yani et al. [7] propose yet another a method to detect entities and their position on triples that have been mentioned in a complex question. Their approach is capable of not only detecting the entity name but also of determining in which triple the entity is located and if the given entity is a head or tail of the triple.

Shin et. al. [15] notice that QA systems suffer notably from the divergence of the unstructured data composing natural language questions and the structured data composing a KG. Existing approaches trying Solve this issue often use lexicons in order to cover differently represented data. Since these lexicons only consider representations for entity and relation mentions the authors propose a new predicate constraint lexicon restricting subject and object types for a predicate. This so-called Predicate Constraints based Question Answering (PCQA from here on) lexicon does not make use of any templates. Rather the authors generated query graphs focusing on matching relations in order to cover diverse types of questions.

Another QA-system proposed by Liang et. al. [5] is based on the idea of splitting the process of translating natural language questions into SPARQL queries into five sub-tasks. First, a random forest model is trained to identify a question's type. Next, various entity recognition and property mapping tools are used to map the question's phrases before all possible triple patterns are created based on these mapped resources. Afterward, possible SPARQL are generated by combining these triple patterns before a Tree-LSTM based ranking model is used to select the most plausible SPARQL query representing the correct intention behind the natural language question. Possible SPARQL queries are then constructed by combining these triple patterns in the query generation step. In order to select the correct SPARQL query among a number of candidate queries for each question, a ranking model based on Tree-LSTM is used in the query ranking step. The ranking model takes into account both the syntactical structure of the question and the tree representation of the candidate queries to select the most plausible SPARQL query representing the correct intention for the respective question.

As this short overview shows, the main tasks in many KGQA systems firstly involve entity/property recognition and matching to respective IRIs in the KG. Secondly, some but not all QA systems proceed by formulating SPARQL queries from these entities. Our following preliminary experiment is therefore tailored to mainly challenge GPT in terms of whether these tasks can be adequately supported by (currently existing) LLMs.

3. Benchmarking LLMs

In order to evaluate the raw performance of large language models we decided to use two of OpenAI's large language models. The first model we used is the GPT 3.5-based ChatGPT. Additionally, we used OpenAI's older GPT 3-based davinci model to give a comparison to ChatGPT's results and to possibly detect structural characteristics of LLMs in the context of question answering.

For this we first let each system/model answer all natural questions directly and secondly indirectly by first generating corresponding SPARQL queries for Wikidata, before subsequently attempting to retrieve their results. This process was done for (i) the student dataset aimed at providing questions that cannot be answered by ChatGPT, (ii) a subset of the SimpleQuestions adaptation for Wikidata [42], and (iii) a subset of the QALD-5 dataset [43], with the student dataset consisting of 14 questions and the two subsets consisting of 15 randomly drawn questions from the original datasets. Extending this study to analogously test further KGQA benchmarks is on our agenda.

We limited our study to the Wikidata KG. This decision has been made for multiple reasons. First, while other popular KGs, such as Freebase, stopped their operations, Wikidata is one of the most popular, and most actively maintained KGs. Besides accounting for the KGs relevancy, this could also mean that Wikidata is better suited to be used when answering information on current events, an expected weakness of LLMs wrt. QA. Secondly, this study aims at uncovering LLMs limitations wrt. KGQA. This renders Wikidata specifically challenging since the task of entity recognition can be assumed to be harder for Wikidata than for other KGs such as DBpedia. This is due Wikidata's numeric identifiers, LLMs should not be able to derive the correct identifiers directly from the question asked, which could be blurred by the inherent semantics of language-based URIs [49] as used for instance in DBpedia. While limiting ourselves to one KG for this preliminary study allowed us to obtain first insights results, expanding and comparing our research wrt. to a comparison with other/multiple KGs is on our agenda.

3.1. Question Answering, Query Generation and Query Execution

All necessary computations and all necessary programming in this study has been done by R scripts [50], using the openai package (version 0.4.0) [51] in combination with OpenAI's API to interact with ChatGPT and davinci. At this point, it must be noted that OpenAI's API allows the usage of different *temperature* options to control how deterministic the behavior of the LLM should be. While the chosen setting might potentially have a significant influence on the results, we used the default temperature setting of 1 in this study to replicate especially ChatGPTs behavior when accessed through its Web interface, to which we have been able to record differences regardless. Secondly, upon trying to execute our scripts with a temperature of 0, supposedly meaning fully deterministic behavior, a later mentioned problem of the LLMs getting stuck at certain questions and repeating previous answers or query structures occurred for most of the questions, rendering the results useless. At this point, we cannot confirm whether this is a result of high server load or if the temperature setting is at least partially responsible for this. Further investigation of the significance of the temperature setting is therefore on our agenda. In this context, note that question 12 and 15 in the sample of the QALD-5 dataset (cf. Table 3) are identical. Since we drew 15 random questions, all having different questions IDs, this indicates that the original dataset contains duplicates. We did deliberately *not* remove those for now, in order determine whether duplicates yield different answers: while slightly different in their wording, the direct natural language answers' content has been identical, and likewise the generated SPARQL queries were identical in our preliminary experimental run. A more in depth investigation in how far repeated "runs" of the experiment yield different results or improvements, also in the context of adapting GPT's temperature parameter, is on our agenda.

In order to generate both the answers to a natural language question (NLQ) and its corresponding SPARQL query the following natural language prompt was used:

> *"Please write me a SPARQL query on Wikidata without comments to answer the following Question: NLQ."*.

Since the used LLMs usually add comments within their queries the passage *"without comments"* has been added to eliminate these comments, which allows for easier processing and subsequently easier execution of these queries. Finally, we used the WikidataQueryServiceR package (version 1.0.0) [52] to execute the generated queries and to retrieve their results.

We note that, besides the elimination of unwanted comments the selected prompt has been carefully designed in a way that aims at preventing the prompt to influence the LLMs answers besides their structural representation: as it was the aim of this preliminary study to determine how LLM's as a standalone option lend themselves to QA and KGQA tasks, we started with a static, uniform prompt. The resulting findings should then form a foundation based on which further insights on the topic of prompt engineering could be derived, i.e., how advanced methods including specifically engineered prompts or hybrid QA approaches that dynamically generate prompts could perform KGQA tasks. While exploring these questions further lies outside of the current study's scope, we consider it a potentially important direction for future work.

3.2. Performance Evaluation

In order to evaluate ChatGPT's and GPT 3's performance both their answers given in natural language and the results of the queries generated by the LLMs have been assigned one out of three possible grades: *Correct*, *Incorrect*, and *No answer*. Correct marks a case where the LLMs were in fact able to answer the given question correctly. Incorrect results mark cases where they were able to answer the question but did so incorrectly. Finally, no answer is assigned to cases where the LLM's where unable to generate an answer to the question asked. We assume that this will be the case when the LLMs are asked about events happening outside of their training period (hence after September 2021).

3.3. Results

In this section, we will show the results generated by ChatGPT and GPT 3 on the student dataset, as well as our subsamples of the SimpleQuestions dataset and the QALD-5 challenge dataset.

3.3.1. Student Dataset

As described earlier, we generated SPARQL queries for each of the questions in the student-generated dataset, retrieved their results as well as the direct answers NLAs) to the NLQs given by ChatGPT and GPT 3, and evaluated them. Table 1 shows the questions in this dataset.

Unsurprisingly, ChatGPT was unable to answer most of the dataset's questions correctly. However, ChatGPT acknowledged its limitations wrt. dates and added a disclaimer at the beginning of its answers stating that its knowledge is limited to dates up to September 2021.[1]

[1]Some further experiments also show that this behavior could seemingly – in the tested GPT versions – sometimes be worked around by prompt reformulation, typically leading to a factually wrong answer.

Table 1
A sample of hard questions for GPT from our student experiment

	Question
1	Who is the current president of the United States?
2	Who won the football worldcup 2022?
3	Give me all Austrian female actors that are aged over 50?
4	Give me all Austrian female actors aged over 50years that are also dancers or singers?
5	When did the famous Brazilian football player Pelé die?
6	For which team does Lionel Messi play?
7	What is the most recent MineCraft Java Edition version?
8	How many people do live on earth?
9	What was the average temperature in Vienna in 2022?
10	Who is the fastest person in the world?
11	What is the oldest painting in the world?
12	Where does the handball world cup take place this year (2023)?
13	Who is CEO of Twitter?
14	Which team won the 'Serie A' championship last season?

Out of 14 questions with our standardized prompt, ChatGPT was able to provide 8 NLAs, out of which three were correct.
As expected, incorrect answers have been given wrt. changes that happened after 2021, such as when ChatGPT was asked to name the "*most recent Minecraft Java Edition version*" (1.19.3 at the time of writing) to which it responded with "*As of September 2021, the most recent version of Minecraft Java Edition is 1.17.1.*".

Another interesting observation is that ChatGPT seemingly can sometimes get "stuck" at a given question. Consider the second and third questions in our student dataset: ChatGPT stated that it is not capable of answering the second question *Who won the football worldcup 2022?* due to it not having taken place by ChatGPT's knowledge. However, ChatGPT gave the same answer to the third question *Give me all Austrian female actors that are aged over 50?*. The same anomaly occurred with question 7 *"What is the most recent MineCraft Java Edition version?* and question 8 *How many people do live on earth?*. We so far did not entirely clarify, whether this behavior was due to an API issue, or due to the sequential nature of the model itself, where different answers are obviously depending on the order of interactions. Regardless of this, ChatGPT was able to generate SPARQL queries for all NLQs within the student dataset. Out of these 14 queries 13 were syntactically correct and could indeed be executed. However, 10 out of these 13 queries returned no results. The three remaining queries returning results have been the queries for question 1 "*Who is the current president of the United States?*" (Listing 1), question 2 "*Who won the football worldcup 2022?*" (Listing 2), and question 8 "*How many people do live on earth?*" (Listing 3).

The former, shown in listing 1 correctly returned Joe Biden, and — looking at the ORDER BY and LIMIT combination, indeed semantically attempts to retrieve the most recent president. We should note though, that question 1 was – as opposed to the other student questions – provided by the instructor upfront, as an example of a question that was correctly translated by GPT, having in mind to find a likely common example question referring to current data, but also probably available verbatim in SPARQL examples that the LLM has been trained upon.

Listing 1: ChatGPT generated query for: Who is the current president of the united states?

```
SELECT ?presidentLabel
WHERE {
    wd:Q30 p:P6 [ ps:P6 ?president ; pq:P580 ?start_time ].
    FILTER NOT EXISTS { ?president p:P582 [] }
    SERVICE wikibase:label { bd:serviceParam
                             wikibase:language "[AUTO_LANGUAGE},en" .}
} ORDER BY DESC(?start_time)
LIMIT 1
```

Listing 2: ChatGPT generated query for: Who won the football worldcup 2022?

```
SELECT ?teamLabel
WHERE {
    ?cup wdt:P31 wd:Q16510064 ;
    wdt:P585 ?date ;
    wdt:P1346 ?team .
    FILTER(YEAR(?date) = 2022)
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
} LIMIT 10
```

Listing 3: ChatGPT generated query for: How many people do live on earth?

```
SELECT (COUNT(?item) as ?count)
WHERE {
        ?item wdt:P31 wd:Q5.
} LIMIT 10
```

On the contrary, the query for question 2, shown in Listing 2, correctly retrieves (somewhat arguably generalizing on the question) winners of sports events in 2022, which could be correct, but among ChatGPTs chosen LIMIT of 10, only persons and no teams (and certainly not the football worldcup winning team) were retrieved. We note that question 2 has been a common example to "challenge" ChatGPT, where earlier incarnations (such as davinci) would answer "Brazil" as a statistically probable, but wrong answer and ChatGPT would, as mentioned above, refuse an answer, hinting on having only training data up to 2021.

Finally, as for question 8 shown in Listing 2, while the query returned was able to return a number, its result of 10546377 missed the target number of ~ 8 billion people by a significant amount and in fact, in this case interestingly, the resulting query simply counts all items that belong to the class human within Wikidata.

In order to test these results for robustness we executed the whole process a second time. In this second run, the before-mentioned problem of ChatGPT getting stuck at a question did not occur at all. Additionally, ChatGPT was now able to answer 11 out of 15, i.e. one additional question, question 11 regarding "*the oldest painting in the world*" now delivered a result. However, the number of correct answers only increased by one.

Surprisingly, ChatGPT's performance wrt. query generation suffered significantly, with the chatbot on the second still being able to generate 15 queries, but out of which only 10 were syntactically correct. This time, only one of these queries (question 1) returned the desired

Listing 4: davinci generated query for: How many people do live on earth?

```
# defaultView:Map
SELECT ?country (SAMPLE(? population) AS ?totalPopulation)
WHERE {
        ?country wdt:P36 wd:Q458.
        ?country wdt:P1082 ?population.
} GROUP BY ?country
HAVING(?totalPopulation > 0)
LIMIT 10
```

results.
An important observation here is that ChatGPT was unable to generate queries for each question when asked manually through its web interface during an initial tryout of the chatbot. At this point, it must be noted that a new OpenAI account, with no prior interactions with ChatGPT, has been used to generate the respective SPARQL queries using the OpenAI API (in order to ensure no bias was added through personalized adaptation on the user account).

Interestingly, upon using the GPT 3-based model davinci (as ChatGPT's predecessor), we were able to observe structural differences between its results and the ones obtained by using ChatGPT.
First, the NLAs generated by davinci did not contain a disclaimer wrt. questions spanning outside of its training time frame. Also, it becomes obvious that ChatGPT has been trained with more recent data than davinci. While ChatGPT was able to correctly answer the question *For which team does Lionel Messi play?* with *Paris Saint-Germain (PSG)*, davinci answered this question with *Barcelona*. We note that overall davinci produced a significantly larger number of factually wrong answers than ChatGPT, which may be partially due to its *outdated* training data, and partially due to the *smaller* training date leading to more "made up" answers. We do note though, that a detailed investigation (comparing factually wrong vs. outdated answers vs. "accidentally right" guesses) in detail is yet on our agenda.

Next, davinci was unable to fully comply with our limitation of not adding any comments to the generated query, while ChatGPT consistently complied with our instructions. Additionally, 11 out of the 14 queries generated by davinci had some sort of syntactical error, making them not executable. See the query for question 8 *How many people do live on earth?* as an example for both of these phenomena:

While this query can simply be syntactically fixed — by removing the whitespace between the ? and the term population within the second line — it still diverts significantly from the original intention of the asked question.

Lastly, the problem of ChatGPT, i.e., getting stuck at a question during its first run, did not occur when using davinci. We assume that this might be related to the amount of traffic ChatGPT experienced while generating the queries, i.e., rather being related to an API problem than the model itself, since this problem does not seem to be consistent in its occurrence and did not occur when using ChatGPT via its web interface.

Overall, davinci's answers appeared, as expected, a lot more arbitrary and outdated than

Table 2
A sample of simple KGQA questions from the SimpleQuestions benchmark [42]

	Question
1	What county is port hadlock in
2	is roll over and play live a hard rock album or an electronica album
3	where does adewale ojomo get his or her nationality from
4	What position does nenad stojaković play?
5	what kinds of movie is appassionata
6	what label is jeanne cherhal signed to
7	where did alberta gay die
8	what kind of film is esterhazy
9	who was albert brooks's mother
10	is tony asher male or female
11	what is a film in the crime fiction genre.
12	What country was the underworld story filmed in
13	where is just married filmed?
14	What type of tv program is the flintstones
15	What football position does siem de jong play

those of the newer ChatGPT model in the preliminary study of our student dataset.

3.3.2. SimpleQuestions

We again generated SPARQL queries for each of the questions in the sample of the SimpleQuestions dataset, retrieved their results as well as the answers to the NLQs given by ChatGPT and davinci and evaluated them.
Table 2 shows the questions listed in this sample of the SimpleQuestions dataset.[2]

The results for this dataset mostly mirror the observations made for the students dataset. However, this time 6 out of 15 answers generated by ChatGPT are results of the chatbot being stuck at a previous question. Out of the 15 generated queries 14 queries were executable but none of them returned any result.
Again, a second run has been done which led to significantly better results. In this second run ChatGPT did not get stuck on a single question and 10 of the provided answers were correct. Similarly, all of the 15 generated queries were syntactically correct, yet still only three out of the 15 queries returned the correct results, while the query for question 14 *What type of tv program is the flintstones* returned a result completely detached from the questions (commune of Italy) which can however be interpreted as one possible answer to question 13 *where is just married filmed?*, hinting at the ChatGPT API getting again stuck during the query generation this time on a previous run.
A possible reason for the comparatively higher success rate observed for this dataset (less emphasis on current or recent data) could be not only its age, with the dataset being unchanged since 2017, but also the general nature of its questions, with many questions being instructions to give an example of something or having many correct possible answers. Take for instance

[2]Please note missing question marks or different capitalization stem from a random sample of the original dataset without any modifications from our side.

Table 3
A sample of simple KGQA questions from the QALD-5 benchmark [43][3]

	Question
1	Who plays Phileas Fogg in the adaptation of Around the World in 80 Days directed by Buzz Kulik?
2	Who succeeded the pope that reigned only 33 days?
3	Which countries have more than ten caves?
4	Which other weapons did the designer of the Uzi develop?
5	A landmark of which city is the home of the Mona Lisa?
6	Was Margaret Thatcher a chemist?
7	Gaborone is the capital of which country member of the African Union?
8	In which country does the Ganges start?
9	Give me a list of all bandleaders that play trumpet.
10	How many missions does the Soyuz programme have?
11	For which movie did the daughter of Francis Ford Coppola receive an Oscar?
12	Are the Rosetta Stone and the Gayer-Andersen cat exhibited in the same museum?
13	What is the ruling party in Lisbon?
14	When were the Hells Angels founded?
15	Are the Rosetta Stone and the Gayer-Andersen cat exhibited in the same museum?

Listing 5: ChatGPT generated query for: How many missions does the Soyuz programme have?

```
SELECT (COUNT(?mission) AS ?countMissions)
WHERE {
    ?mission wdt:P31 wd:Q209343.
    ?mission wdt:P361 wd:Q127846.
} LIMIT 10
```

question 11 *what is a film in the crime fiction genre*. With numerous crime fiction films existing and their success or popularity not being a limitation stated in the question it can be assumed that ChatGPT should be able to name at least one item fitting the definition of being a film and belonging to the genre crime fiction. From this, we can expect ChatGPT to be able to answer most questions in this dataset correctly that do not ask for elements subject to change, such as the CEO of a company or which player currently plays for a certain team.

3.3.3. QALD-5

Lastly, we analogously again generated SPARQL queries for each of the questions in a sample of the QALD-5 dataset (shown in Table 3), retrieved their results, and analyzed direct answers to the NLQs given by ChatGPT and davinci.

When directly answering the questions in this subset of QALD-5, ChatGPT did not get stuck on any of the 15 questions and was able to answer *9* of the 15 questions correctly. However, while ChatGPT was able to generate syntactically correct queries for all of the 15 questions, the only query returning a result was the query for question 10 *How many missions does the Soyuz programme have?*. Yet, the only element of this query closely related to the actual question is the count function.

[3]Note: the duplicate question 12+15 were discussed in Section 3.1.

3.4. Summary of Results

Summarizing the results of our initial experiments, overall, we admittedly are only at the start of our research. Yet, we have already gained valuable insights into the potential and gaps when trying to leverage LLMs for (factual) question answering, with a focus on questions the answers of which should be retrievable from KGs.

How does LLM-based QA differ from established KGQA approaches and what are the respective strengths, weaknesses, and challenges of the two methods?

While LLMs show promising results wrt. QA it became clear that these models are limited by various factors. Most importantly, both ChatGPT's and davinci's limitations in direct question answering were mostly related to outdated training data, such that they performed particularly well on older QA benchmarks.

Unsurprisingly, the newer ChatGPT model performed significantly better on both the direct question answering tasks and also in particular in terms of the syntactical correctness of queries; we may expect further significant advances in the just released GPT4 model.

Additionally, some unexpected behaviors resulted from inexplicable effects of interacting with the OpenAI APIs', in terms of order-dependent answers that appeared to be actually "stuck" answers to prior queries. Unfortunately, we could not yet determine whether these were related to simple API bugs or due to the model; however really open LLMs would certainly allow investigating order-dependency or alike in a much more transparent manner, than OpenAI's current, closed business model that in fact may switch to a paid only approach.

A summary of both ChatGPT's and davinci's results wrt. direct question answering can be found in Tables 5 and 6.

In terms of query formulation, ChatGPT produced a high share of syntactically correct queries, but very few reflecting the actual question; we do hypothesize that this is largely due to a lack of explicit entity recognition, i.e., recognizing correct relevant IRIS (i.e., in the case of Wikidata relevant Q- and P-identifiers of entities and properties. A more in depth analysis of the resulting queries in terms of *semantic distance* of the extracted identifiers, or investigating in how far LLMS can be used for supporting the entity recognition subtask in isolation as part of a KGQA pipeline is on our agenda.

Especially for the latter point, one has to assume that correct query formulations so far rather stem from verbatim SPARQL examples in the training data for common questions than from an actual understanding of the entities and query structure. We may still assume that the quality of such queries will improve in the future, even now already we encountered hardly any syntactical errors.

A summary of ChatGPT's results wrt. query generation can be found in Table 7.

Which components used in KGQA-systems could be enhanced using LLMs?

While the LLMs in questions showed mediocre results by themselves they potentially inherit the capabilities to improve already existing QA-approaches. We believe that LLMs could provide especially useful in the task of entity recognition which forms part of many existing KGQA-systems. Using LLMs to find synonyms for words occurring in the question asked, extracting the questions underlying meaning, and using them in combination with query

generation templates or by implementing extensive prompt engineering to give the LLMs hints on how to structure their queries.

What types of questions are found in existing benchmarks for KGQA approaches and in how far can these be used in benchmarking LLM-based QA approaches?

While different benchmarks use different categories to categorize their questions some studies provide a holistic categorization of the questions and queries provided in different benchmark datasets. A summary of the used (sub)datasets questions categorized in accordance to CBench's wh-questions classification [1] can be found in Table 4 while Tables 5 and 6 provide a summary of the correctly answered questions by their type.

The results of our study show that LLMs have a particularly hard time answering questions forming some type of count or, resp., asking for *all* entities of a certain category (particularly though, in terms of KGQA also because knowledge in common KGs is typically incomplete). Aside from this, LLMs struggle to answer questions including recent events due to their limited training period.

An additional analysis wrt. to the questions' categories and patterns in the LLMs' results will be conducted in future work.

How can a comprehensive benchmark for LLM-based, KGQA-based but also combined QA approaches be constructed that is challenging the current weaknesses of both approaches?

While it became obvious that a comprehensive benchmark must contain questions aimed at recent/current events, these types of questions are not only harder to fact-check but inherit additional complications due to the resulting need of constant adaption of the benchmark. Additionally, a comprehensive KGQA benchmark must include questions that require the KGQA-system to form some sort of arithmetic or logical linking, such as counting entities related to a word in the asked question, etc.

4. Conclusion

In our preliminary study, we analyzed the performance of two of OpenAI's large language models davinci (GPT 3) and ChatGPT (GPT 3.5) against a set of questions established by students and two subsets of the established benchmark datasets SimpleQuestions and QALD-5. We used both models to answer the questions in each dataset, as well as to generate SPARQL queries aimed at retrieving these answers from the knowledge base Wikidata. Our results demonstrate the limitations of large language models, which mainly lies in their training time frame as well as their stability. Additionally, we show that LLMs are in principle capable of generating functioning queries. While being able to consistently generate structurally and syntactically correct queries, they however demonstrate bad performance wrt. entity detection, resulting in the generated queries not returning the desired results. Therefore, the question remains open how large language models can be used in combination with existing question answering

systems and specifically how existing approaches can be used to substitute the LLM's deficits regarding entity detection. The presented preliminary paper comprises the first results of a recently started master thesis project. Starting from these initial insights, we look forward to discussing routes ahead at the workshop and collecting feedback for our ongoing experiments.

5. Tables

	Student	SimpleQuestions	QALD-5
What	3	9	1
When	1	0	1
Where	1	3	0
Which	2	0	6
Who	4	1	2
Whom	0	0	0
Whose	0	0	0
How	1	0	1
Yes/No	0	0	3
Requests	2	0	1
Topical	0	2	0
Sum	14	15	15

Table 4
Distribution of wh-question types.

	Student	SimpleQuestions	QALD-5
What	1	5	1
When	0	0	1
Where	0	2	0
Which	1	0	4
Who	2	1	1
Whom	0	0	0
Whose	0	0	0
How	0	0	0
Yes/No	0	0	2
Requests	0	0	0
Topical	0	2	0
Sum	4	10	9

Table 5
Correctly answered questions GPT 3.5.

	Student	SimpleQuestions	QALD-5
What	0	4	0
When	0	0	1
Where	0	1	0
Which	1	0	4
Who	1	1	0
Whom	0	0	0
Whose	0	0	0
How	0	0	0
Yes/No	0	0	2
Requests	0	0	0
Topical	0	0	0
Sum	2	6	7

Table 6
Correctly answered questions GPT 3.

	Student	SimpleQuestions	QALD-5
Generated queries	14	15	15
Syntactically correct	10	15	15
Syntactically incorrect	4	0	0
Correct	1	3	0
Incorrect	1	1	1
No answer	8	11	14

Table 7
Results for generated queries GPT 3.5.

References

[1] A. Orogat, I. Liu, A. El-Roby, Cbench: Towards better evaluation of question answering over knowledge graphs, 2021. URL: https://arxiv.org/abs/2105.00811. doi:10.48550/ARXIV.2105.00811.

[2] D. Diefenbach, A. Both, K. Singh, P. Maret, Towards a question answering system over the semantic web, 2018. URL: https://arxiv.org/abs/1803.00832. doi:10.48550/ARXIV.1803.00832.

[3] A. Dhandapani, V. Vadivel, Question answering system over semantic web, IEEE Access 9 (2021) 46900–46910. URL: https://doi.org/10.1109/access.2021.3067942. doi:10.1109/access.2021.3067942.

[4] S. Vakulenko, J. D. F. Garcia, A. Polleres, M. de Rijke, M. Cochez, Message passing for complex question answering over knowledge graphs, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, ACM, 2019. URL: https://doi.org/10.1145/3357384.3358026. doi:10.1145/3357384.3358026.

[5] S. Liang, K. Stockinger, T. M. de Farias, M. Anisimova, M. Gil, Querying knowledge graphs in natural language, Journal of Big Data 8 (2021). URL: https://doi.org/10.1186/s40537-020-00383-w. doi:10.1186/s40537-020-00383-w.

[6] H. Cui, T. Peng, L. Feng, T. Bao, L. Liu, Simple question answering over knowledge graph enhanced by question pattern classification, Knowledge and Information Systems 63 (2021) 2741–2761. URL: https://doi.org/10.1007/s10115-021-01609-w. doi:10.1007/s10115-021-01609-w.

[7] M. Yani, A. A. Krisnadhi, I. Budi, A better entity detection of question for knowledge graph question answering through extracting position-based patterns, Journal of Big Data 9 (2022). URL: https://doi.org/10.1186/s40537-022-00631-1. doi:10.1186/s40537-022-00631-1.

[8] M. Zaib, W. E. Zhang, Q. Z. Sheng, A. Mahmood, Y. Zhang, Conversational question answering: a survey, Knowledge and Information Systems 64 (2022) 3151–3195. URL: https://doi.org/10.1007/s10115-022-01744-y. doi:10.1007/s10115-022-01744-y.

[9] C. Mercer, The rise of chat gpt: The future of conversational ai, https://medium.com/@conan.mercer/the-rise-of-chat-gpt-the-future-of-conversational-ai-91622b9db303, 2023. Accessed on March 06, 2023.

[10] S. Garg, ChatGPT alternatives that will blow your mind in 2023 — writesonic.com, https://writesonic.com/blog/chatgpt-alternatives/, 2023. Accessed on March 06, 2023.

[11] L. Jiang, R. Usbeck, Knowledge graph question answering datasets and their generalizability, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2022. URL: https://doi.org/10.1145/3477495.3531751. doi:10.1145/3477495.3531751.

[12] F. A. Acheampong, H. Nunoo-Mensah, W. Chen, Transformer models for text-based emotion detection: a review of BERT-based approaches, Artificial Intelligence Review 54 (2021) 5789–5829. URL: https://doi.org/10.1007/s10462-021-09958-2. doi:10.1007/s10462-021-09958-2.

[13] Z. Abbasiantaeb, S. Momtazi, Entity-aware answer sentence selection for question answering with transformer-based language models, Journal of Intelligent Information Systems 59 (2022) 755–777. URL: https://doi.org/10.1007/s10844-022-00724-6. doi:10.1007/

s10844-022-00724-6.

[14] G. Mai, K. Janowicz, L. Cai, R. Zhu, B. Regalia, B. Yan, M. Shi, N. Lao, iSE-KGE/i : A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting, Transactions in GIS 24 (2020) 623–655. URL: https://doi.org/10.1111/tgis.12629. doi:10.1111/tgis.12629.

[15] S. Shin, X. Jin, J. Jung, K.-H. Lee, Predicate constraints based question answering over knowledge graph, Information Processing &amp Management 56 (2019) 445–462. URL: https://doi.org/10.1016/j.ipm.2018.12.003. doi:10.1016/j.ipm.2018.12.003.

[16] J. Gomes, R. C. de Mello, V. Ströele, J. F. de Souza, A study of approaches to answering complex questions over knowledge bases, Knowledge and Information Systems 64 (2022) 2849–2881. URL: https://doi.org/10.1007/s10115-022-01737-x. doi:10.1007/s10115-022-01737-x.

[17] P. Do, T. H. V. Phan, Developing a BERT based triple classification model using knowledge graph embedding for question answering system, Applied Intelligence 52 (2021) 636–651. URL: https://doi.org/10.1007/s10489-021-02460-w. doi:10.1007/s10489-021-02460-w.

[18] W. Jin, B. Zhao, H. Yu, X. Tao, R. Yin, G. Liu, Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning, Data Mining and Knowledge Discovery 37 (2022) 255–288. URL: https://doi.org/10.1007/s10618-022-00891-8. doi:10.1007/s10618-022-00891-8.

[19] R. Wang, M. Wang, J. Liu, M. Cochez, S. Decker, Structured query construction via knowledge graph embedding, Knowledge and Information Systems 62 (2019) 1819–1846. URL: https://doi.org/10.1007/s10115-019-01401-x. doi:10.1007/s10115-019-01401-x.

[20] U. Sawant, S. Garg, S. Chakrabarti, G. Ramakrishnan, Neural architecture for question answering using a knowledge graph and web corpus, Information Retrieval Journal 22 (2019) 324–349. URL: https://doi.org/10.1007/s10791-018-9348-8. doi:10.1007/s10791-018-9348-8.

[21] H. Jung, W. Kim, Automated conversion from natural language query to SPARQL query, Journal of Intelligent Information Systems 55 (2020) 501–520. URL: https://doi.org/10.1007/s10844-019-00589-2. doi:10.1007/s10844-019-00589-2.

[22] D. Diefenbach, V. Lopez, K. Singh, P. Maret, Core techniques of question answering systems over knowledge bases: a survey, Knowledge and Information Systems 55 (2017) 529–569. URL: https://doi.org/10.1007/s10115-017-1100-y. doi:10.1007/s10115-017-1100-y.

[23] S. Kafle, N. de Silva, D. Dou, An overview of utilizing knowledge bases in neural networks for question answering, Information Systems Frontiers 22 (2020) 1095–1111. URL: https://doi.org/10.1007/s10796-020-10035-2. doi:10.1007/s10796-020-10035-2.

[24] Y.-M. Kim, T.-H. Lee, S.-O. Na, Constructing novel datasets for intent detection and ner in a korean healthcare advice system: guidelines and empirical results, Applied Intelligence 53 (2022) 941–961. URL: https://doi.org/10.1007/s10489-022-03400-y. doi:10.1007/s10489-022-03400-y.

[25] E. Erdem, M. Kuyu, S. Yagcioglu, A. Frank, L. Parcalabescu, B. Plank, A. Babii, O. Turuta, A. Erdem, I. Calixto, E. Lloret, E.-S. Apostol, C.-O. Truică, B. Šandrih, S. Martinčić-Ipšić, G. Berend, A. Gatt, G. Korvel, Neural natural language generation: A survey on multilinguality, multimodality, controllability and learning, Journal of Artificial Intelligence Research 73 (2022) 1131–1207. URL: https://doi.org/10.1613/jair.1.12918.

doi:10.1613/jair.1.12918.
[26] T. Adewumi, F. Liwicki, M. Liwicki, State-of-the-art in open-domain conversational AI: A survey, Information 13 (2022) 298. URL: https://doi.org/10.3390/info13060298. doi:10.3390/info13060298.
[27] D. M. Korngiebel, S. D. Mooney, Considering the possibilities and pitfalls of generative pre-trained transformer 3 (GPT-3) in healthcare delivery, npj Digital Medicine 4 (2021). URL: https://doi.org/10.1038/s41746-021-00464-x. doi:10.1038/s41746-021-00464-x.
[28] Y. Matveev, O. Makhnytkina, P. Posokhov, A. Matveev, S. Skrylnikov, Personalizing hybrid-based dialogue agents, Mathematics 10 (2022) 4657. URL: https://doi.org/10.3390/math10244657. doi:10.3390/math10244657.
[29] Y. Yang, J. Cao, Y. Wen, P. Zhang, Multiturn dialogue generation by modeling sentence-level and discourse-level contexts, Scientific Reports 12 (2022). URL: https://doi.org/10.1038/s41598-022-24787-1. doi:10.1038/s41598-022-24787-1.
[30] Z. Ahmad, A. Ekbal, S. Sengupta, P. Bhattacharyya, Neural response generation for task completion using conversational knowledge graph, PLOS ONE 18 (2023) e0269856. URL: https://doi.org/10.1371/journal.pone.0269856. doi:10.1371/journal.pone.0269856.
[31] D. Jannach, L. Chen, Conversational recommendation: A grand AI challenge, AI Magazine 43 (2022) 151–163. URL: https://doi.org/10.1002/aaai.12059. doi:10.1002/aaai.12059.
[32] S. Huh, Are ChatGPT's knowledge and interpretation ability comparable to those of medical students in korea for taking a parasitology examination?: a descriptive study, Journal of Educational Evaluation for Health Professions 20 (2023) 1. URL: https://doi.org/10.3352/jeehp.2023.20.01. doi:10.3352/jeehp.2023.20.01.
[33] G. Caldarini, S. Jaf, K. McGarry, A literature survey of recent advances in chatbots, Information 13 (2022) 41. URL: https://doi.org/10.3390/info13010041. doi:10.3390/info13010041.
[34] V. Shankar, S. Parsana, An overview and empirical comparison of natural language processing (NLP) models and an introduction to and empirical application of autoencoder models in marketing, Journal of the Academy of Marketing Science 50 (2022) 1324–1350. URL: https://doi.org/10.1007/s11747-022-00840-3. doi:10.1007/s11747-022-00840-3.
[35] N. Alswaidan, M. E. B. Menai, A survey of state-of-the-art approaches for emotion recognition in text, Knowledge and Information Systems 62 (2020) 2937–2987. URL: https://doi.org/10.1007/s10115-020-01449-0. doi:10.1007/s10115-020-01449-0.
[36] S.-E. Kim, Y.-S. Lim, S.-B. Park, Strong influence of responses in training dialogue response generator, Applied Sciences 11 (2021) 7415. URL: https://doi.org/10.3390/app11167415. doi:10.3390/app11167415.
[37] N. Tsinganos, P. Fouliras, I. Mavridis, Applying BERT for early-stage recognition of persistence in chat-based social engineering attacks, Applied Sciences 12 (2022) 12353. URL: https://doi.org/10.3390/app122312353. doi:10.3390/app122312353.
[38] H. Snyder, Literature review as a research methodology: An overview and guidelines, Journal of Business Research 104 (2019) 333–339. URL: https://doi.org/10.1016/j.jbusres.2019.07.039. doi:10.1016/j.jbusres.2019.07.039.
[39] A. P. L. J. J. S. A. K. C. M. J. H. J. R. A.-C. N. N. M. S. A. B. Ricardo Usbeck, Xi Yan, Qald-10 - the 10th challenge on question answering over linked data, 2023. URL: https://semantic-web-journal.net/content/qald-10-%E2%80%

94-10th-challenge-question-answering-over-linked-data-0, under submission.
[40] M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia, in: Proceedings of the 18th International Semantic Web Conference (ISWC), Springer, 2019.
[41] P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann, Lc-quad: A corpus for complex question answering over knowledge graphs, in: Proceedings of the 16th International Semantic Web Conference (ISWC), Springer, 2017, pp. 210–218.
[42] D. Diefenbach, T. P. Tanon, K. D. Singh, P. Maret, Question answering benchmarks for wikidata, in: Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017., 2017. URL: http://ceur-ws.org/Vol-1963/paper555.pdf.
[43] C. Unger, C. Forescu, V. Lopez, A.-C. Ngonga Ngomo, E. Cabrio, P. Cimiano, S. Walter, Question answering over linked data (qald-5), 2015.
[44] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative knowledgebase, Commun. ACM 57 (2014) 78–85. URL: https://doi.org/10.1145/2629489. doi:10.1145/2629489.
[45] V. Lopez, C. Unger, P. Cimiano, E. Motta, Evaluating question answering over linked data, Web Semantics Science Services And Agents On The World Wide Web 21 (2013) 3–13. doi:10.1016/j.websem.2013.05.006.
[46] A. Bordes, N. Usunier, S. Chopra, J. Weston, Large-scale simple question answering with memory networks, 2015. URL: https://arxiv.org/abs/1506.02075. doi:10.48550/ARXIV.1506.02075.
[47] R. F. Jonathan Berant, Andrew Chou, P. Liang, Semantic parsing on freebase from question-answer pairs, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (2013) 1533–1544.
[48] D063520, Github - wdaquacore0questions, 2017. URL: https://github.com/WDAqua/WDAquaCore0Questions, accessed on February 28, 2023.
[49] S. de Rooij, W. Beek, P. Bloem, F. van Harmelen, S. Schlobach, Are names meaningful? quantifying social meaning on the semantic web, in: P. Groth, E. Simperl, A. J. G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck, Y. Gil (Eds.), The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I, volume 9981 of *Lecture Notes in Computer Science*, 2016, pp. 184–199. URL: https://doi.org/10.1007/978-3-319-46523-4_12. doi:10.1007/978-3-319-46523-4_12.
[50] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2022. URL: https://www.R-project.org/.
[51] I. Rudnytskyi, openai: R Wrapper for OpenAI API, 2023. URL: https://CRAN.R-project.org/package=openai, r package version 0.4.0.
[52] M. Popov, WikidataQueryServiceR: API Client Library for 'Wikidata Query Service', 2020. URL: https://CRAN.R-project.org/package=WikidataQueryServiceR, r package version 1.0.0.