StructGPT: A General Framework for Large Language Model to Reason over Structured Data

Jinhao Jiang[1,3]*, **Kun Zhou**[2,3]*, **Zican Dong**[1], **Keming Ye**[4], **Wayne Xin Zhao**[1,3]† and **Ji-Rong Wen**[1,2,3]

[1]Gaoling School of Artificial Intelligence, Renmin University of China.

[2]School of Information, Renmin University of China.

[3]Beijing Key Laboratory of Big Data Management and Analysis Methods.

[4]University of Electronic Science and Technology of China.

{jiangjinhao,jrwen}@ruc.edu.cn,francis_kun_zhou@163.com

{210708dzc,kemingye2002,batmanfly}@gmail.com

Abstract

In this paper, we study how to improve the zero-shot reasoning ability of large language models (LLMs) over structured data in a unified way. Inspired by the study on tool augmentation for LLMs, we develop an *Iterative Reading-then-Reasoning (IRR)* approach for solving question answering tasks based on structured data, called **StructGPT**. In our approach, we construct the specialized function to collect relevant evidence from structured data (*i.e., reading*), and let LLMs concentrate the reasoning task based on the collected information (*i.e., reasoning*). Specially, we propose an *invoking-linearization-generation* procedure to support LLMs in reasoning on the structured data with the help of the external interfaces. By iterating this procedures with provided interfaces, our approach can gradually approach the target answer to a given query. Extensive experiments conducted on three types of structured data demonstrate the effectiveness of our approach, which can significantly boost the performance of ChatGPT and achieve comparable performance against the full-data supervised-tuning baselines. Our codes and data are publicly available at https://github.com/RUCAIBox/StructGPT.

1 Introduction

Recently, large language models (LLMs) (Brown et al., 2020; Chowdhery et al., 2022; Zhao et al., 2023) have made remarkable advancements in the field of natural language processing (NLP). Existing research (Ouyang et al., 2022; Zhang et al., 2022; Tay et al., 2022) has demonstrated that LLMs (*e.g.,* ChatGPT [1] or GPT-4 (OpenAI, 2023)) have strong zero-shot and few-shot capacities to solve a broad range of tasks by feeding specially designed prompts, without task-specific fine-tuning.

Despite the success, recent work has also revealed that LLMs may generate unfaithful information in conflict with the factual knowledge (Bang et al., 2023), and also fall short of mastering domain-specific or real-time knowledge (Schick et al., 2023; Peng et al., 2023; Jiang et al., 2022a). A direct solution to the above issues is to augment LLMs with external knowledge resources, so as to amend the incorrect generations. Among these resources, structured data (*e.g.,* knowledge graphs and databases), has been widely used as the carrier of the required knowledge for LLMs. Unlike plain text, structured data is organized in a standardized format, conforming to some data model. For example, knowledge graphs (KGs) are often organized as fact triples that state the relations between head entities and tail entities, and data tables are organized in the form of column-indexed records by rows. However, as structured data has special data formats or schemas that LLMs have not seen during pre-training, they may be not fully grasped or understood by LLMs (Wei et al., 2021). A straightforward way to solve this problem is to linearize the structured data into a sentence that LLMs can well understand. While, the amount of structured data is often vast, making it infeasible to include all the data records in the input prompt (*e.g.,* a maximum context length of 4096 for ChatGPT [2]).

Regarding the above challenges, we are inspired by the tool manipulation strategy for augmenting the abilities of LLMs (Schick et al., 2023; Nakano et al., 2021; Gao et al., 2022b). The basic idea of our approach is to incorporate specialized interfaces (*e.g.,* extracting columns for tables) to manipulate the structured data records. With these interfaces, we can effectively reduce the search space of the data records, and more accurately identify the required evidence to fulfill specific tasks. In this way, LLMs can concentrate on reasoning based on the evidence obtained from the interfaces. To im-

* Equal contributions.

† Corresponding authors.

[1]https://chat.openai.com/

[2]https://platform.openai.com/docs/models/gpt-3-5

plement the interface-augmented approach, there remain two key problems, namely how to design suitable interfaces for specific tasks and how to utilize them for reasoning by LLMs, which are the focus of this work.

To this end, in this paper, we propose an *Iterative Reading-then-Reasoning (IRR)* approach for solving the tasks based on structured data, namely **StructGPT**. Our approach considers two major functions to fulfill different tasks, namely collecting relevant evidence (*reading*) and inferring the answer or planning subsequent steps (*reasoning*). In essence, we disentangle the two processes of *reading* and *reasoning* for LLMs: we utilize the interface of structured data to implement accurate, efficient data access and filtering, and further leverage the reasoning ability of LLMs to figure out the next step or the final result for the question. Specially, we propose an *invoking-linearization-generation* procedure to support LLMs in reading and reasoning on the structured data with the help of the external interfaces. By iterating this procedure with provided interfaces, we can gradually approach the target answer to a given question.

To our knowledge, this is the first work that explores how to support LLMs in reasoning on multiple types of structured data (including tables, KGs, and DBs) in a unified paradigm. To evaluate the effectiveness of our approach, we conduct extensive experiments on a wide-range of tasks (*e.g.,* KG-based question answering, Table-based question answering, and DB-based Text-to-SQL). Experimental results on 8 datasets demonstrate that our proposed approach can effectively enhance the reasoning performance of ChatGPT on structured data, even comparable with competitive full-data supervised-tuning methods.

• *KGQA*. In KGQA task, our approach yields an increase of 11.4% of Hits@1 on WebQSP. In multi-hop KGQA datasets (*e.g.,* MetaQA-2hop and MetaQA-3hop), with the help of our approach, ChatGPT's performance can be improved by up to 62.9% and 37.0%.

• *TableQA*. In TableQA task, compared to directly using ChatGPT, our approach improves denotation accuracy by approximately 3% - 5% in WTQ and WikiSQL. In table fact verification, our approach improves accuracy by 4.2% in TabFact.

• *Text-to-SQL*. In Text-to-SQL task, compared to directly using ChatGPT, our approach improves execution accuracy by approximately 4% across three datasets.

2 Preliminary

In this section, we introduce the definition of structured data, and then present the problem statement.

Structured Data. Structured data (*e.g.,* data tables and knowledge graphs) refers to the data that is in a standardized format, conforming to some data model (Xie et al., 2022; Chen et al., 2009). Due to the formal structure, it is easy and efficient to access and query structured data using formal languages (*e.g.,* SQL and SPARQL for databases) or specific algorithms (*e.g.,* triples search for knowledge graphs). In this work, we mainly focus on three types of structured data, namely knowledge graphs (KG), data tables (Table), and databases (DB), as they play an important role as the knowledge source in helping solve complex reasoning tasks, described as follows.

• *Knowledge Graph.* A knowledge graph (KG) is composed of a large number of formatted triples to store the factual knowledge, denoted as $\mathcal{G} = \{\langle e, r, e' \rangle | e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the set of entities and relations, respectively. A triple $\langle e, r, e' \rangle$ represents the fact that there is a relation r between the head entity e and the tail entity e'.

• *Data Table.* A data table $\mathcal{T}$ (*table* in short) contains multiple columns $\{c_i\}_{i=1}^{C}$ and rows $\{l_j\}_{j=1}^{R}$, where each row l_j denotes a data record formatted by the attributes indexed by columns $\{c_i\}_{i=1}^{C}$, and $v_{i,j}$ denotes the content in the cell corresponding to the position at column i and row j.

• *Database.* A database (DB) is a collection of structured data. Typically, it consists of N data tables, denoted as $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$. Besides the column names, the foreign keys across all tables are also available to link the data from two tables, denoted as $\{(c_i^{(k)}, c_j^{(h)})\}$, where $c_i^{(k)}$ and $c_j^{(h)}$ denote the i-th and j-th columns in the k-th and h-th tables, respectively.

Problem Statement. This work mainly focuses on using large language models (LLMs) to solve complex reasoning tasks based on structured data. Formally, it can be described as a question answering task: given a natural language question q and an accessible structured data $\mathcal{S}$ (*e.g.,* a knowledge graph or database), the LLM needs to extract useful evidence from $\mathcal{S}$ and then generates the expected result to answer the question q based on the extracted evidence. According to the task requirement, the generated result can be either free-form answers in natural language or structured experessions (*e.g.,* SQL statements) to be executed for obtaining the answer from $\mathcal{S}$. Since we consider three types of structured data (Section 3), our tasks can be instantiated as follows:

- KG based question answering (KGQA)
- Table based question answering (TableQA)
- DB based semantic parsing (Text-to-SQL)

3 Approach

In this section, we present the proposed *Iterative Reading-then-Reasoning (IRR)* approach for question answering tasks based on structured data.

3.1 Overview

In this work, we assume that LLMs have to rely on the evidence contained in the structured data to solve the three tasks described in Section 2. An intuitive idea is to conduct a two-stage procedure as prior studies on retrieval-augmented approaches (Izacard et al., 2022; Oguz et al., 2022), in which LLMs are employed to first collect sufficient evidence relating to the question and then figure out the answer by the LLMs. However, such an approach is not directly applicable for structured data. Although LLMs are capable of solving diverse tasks in natural language, they have limited capacities in accurately representing and understanding structured data, especially for their contained domain-specific knowledge (Moiseev et al., 2022; Emelin et al., 2022).

To address this difficulty, our solution is inspired by the use of specialized tools in solving complex tasks for LLMs (Nakano et al., 2021; Gao et al., 2022b; Schick et al., 2023). We are noted that structured data is well organized and supports easy access via formal language or queries (called *interface* for generality). The basic idea of our approach is to disentangle the two processes of *reading* and *reasoning* for LLMs: we utilize the interface of structured data to implement accurate, efficient data access and filtering (*obtaining the relevant evidence*), and further utilize the reasoning ability of LLMs to figure out the final plan or result for the question (*fulfilling the task*). In this way, LLMs can concentrate on the reasoning process in answering the question, without considering the specialized approach to reading the structured data.

Specially, in our framework, we encapsulate the structured data as a black-box system, and provide specific interfaces for LLMs to access the contained data. Further, we propose an invoking-linearization-generation procedure that enables LLMs to read and extract useful evidence from structured data via the corresponding interface. By iterating the above procedure with provided interfaces, we can gradually approach the target answer by leveraging the superior reasoning ablities of LLMs.

In what follows, we introduce the key points of our approach in detail.

3.2 Interfaces for Structured Data

Due to the standardized data formats, structured data is often equipped with efficient data management ways, *e.g.,* SQL for database. In our approach, we aim to provide LLMs with specialized interfaces based on them, helping LLMs to *read* and *utilize* the structured data. Next, we present the specially designed interfaces for KG, table, and DB.

Interfaces for Knowledge Graph. When performing complex reasoning on a KG, existing work (He et al., 2021; Sun et al., 2018) typically starts from a certain entity (about the question topic), and jumps along with the relations until reaching the answer. In this process, LLMs should be aware of the neighboring relations of the current entity, and the neighboring triples with certain relations to the current entity. Based on it, LLMs can select the relevant relations and triples from them to finally reach the answer entities. For this purpose, we devise two functions for assisting LLMs to accomplish the above operations.

- *Extract_Neighbor_Relations* (e): extracts all the neighboring relations of the entity e.
- *Extract_Triples* $(e, \{r\})$: extracts all the triples with the relation in $\{r\}$ and head entity e.

Interfaces for Tables. Given a data table, LLMs need to know its contained column names, and can access the content by row or column, enabling LLMs to extract the sub-table containing relevant columns and rows from it. Therefore, we define three functions:

- *Extract_Column_Name* $(\mathcal{T})$: extracts all the column names of a table $\mathcal{T}$.
- *Extract_Columns* $(\mathcal{T}, \{c\})$: extracts the contents of columns from a table $\mathcal{T}$ by indices $\{c\}$.
- *Extract_SubTable* $(\mathcal{T}, \{c\}, \{j\})$: extracts the sub-table specified by the column indices $\{c\}$ and row indices $\{j\}$ from a table $\mathcal{T}$.

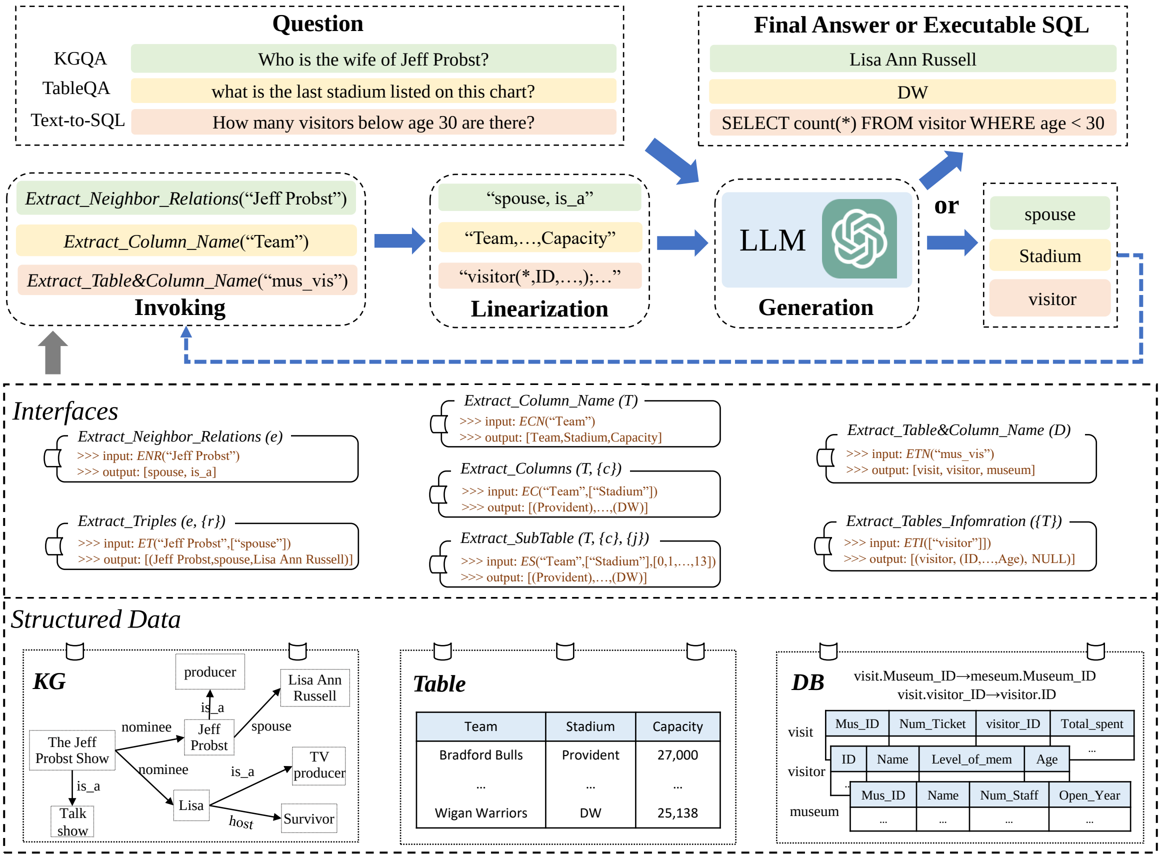

Figure 1: The overview of the proposed iterative reading-then-reasoning approach. We design specialized interfaces for reading structured data, and iterate the invoking-linearization-generation procedure to utilize LLMs for performing reasoning on the interfaces, until deriving the final answer or executable SQL.

Interfaces for Database. Considering a simplified setting when querying the database, LLMs should be aware of all the contained tables and columns (by name) for relevant tables selection, and can also acquire the detailed columns and foreign key information from the selected tables to search the answer. Thus, we devise two functions as follows:

• *Extract_Table&Column_Name* ($\mathcal{D}$): extracts the names of all the tables and their contained columns from the database.

• *Extract_Tables_Information* ($\{\mathcal{T}\}$): extracts the table names, column names, and foreign keys from a set of tables $\{\mathcal{T}\}$.

3.3 Inferface-augmented Reasoning

Based on the above interfaces, we propose a general *invoking-linearization-generation* procedure that can be iterated in multiple turns for utilizing LLMs to perform reasoning on structured data. For each iteration, based on the currently collected data, we first invoke an interface to extract relevant evidence from structured data, then linearize it into textual prompt, and finally feed the prompt into the LLM for generation (selecting useful data or predicting the answer).

Invoking an Interface. In this step, we aim to invoke an interface for extracting the relevant information from the structured data. According to the design of interfaces in Section 3.2, we construct the input based on the currently available data (*e.g.,* entity and table), and then invoke the interface to obtain more detailed relevant information (*e.g.,* neighboring relations and column names), which will be fed into LLMs for collecting useful information or generating the answer.

Information Linearization. Based on the extracted information, it is essential to convert it into a textual sentence that can be understood by LLMs. For the information from KG (*i.e.,* relations and triples), we directly concatenate them into a long sentence marked by specific separation and boundary symbols. For table and database, we leverage the same way to linearize the extracted table names or column names. While for contents in columns and rows, we follow existing work (Pasupat and Liang, 2015) that first converts them into triples, where head entities are the row indices, relations are column names and tail entities are the content in the cell, *e.g.,* "*(row 1, year, 1896)*" and "*(row 1, city, Athens)*". Then, for each row, we extract the row indice in the front and omit it in the triples, to compose a simplified sentence, *e.g.,* "*row 1: (year, 1896), (city, Athens)*". For multiple rows, we concatenate them into a long sentence via a special separation symbol.

LLM for Generation. After linearization, we further compose the input prompt for LLMs. Specially, we design two types of prompts to fulfill different purposes[3]:

• The first type of prompts mostly adopts the following pattern: "*Here are [Y]. Which [X] are most relevant to answer the question [Q]*". It aims to elicit the abilties of LLMs to select useful evidence (*i.e., [X]*) from linearized extracted information (*i.e., [Y]*), according to the question (*i.e., [Q]*).

• The second type of prompts follow the pattern: "*Based on [Y], please generate [Z] for the question [Q]*". It aims to predict the targeted results (*i.e., [Z]*) for the given question (*i.e., [Q]*) based on the linearized extracted information (*i.e., [Y]*). Note that the targeted results can be either the answer string or executable formal language (*e.g.,* SQL) that can lead to the final answer.

3.4 Iterative Reasoning with Interfaces

By iterating the above invoking-linearization-generation procedure on designed interfaces, LLMs can progressively capture more useful evidence for deriving the final answer. In the following, we describe the instances of the above general workflow for the tasks described in Section 2, since they deal with very different structured data and vary in the task settings.

KG-based Question Answering (KGQA). This task aims to find the answer entities for the question based on the KG. Following existing work (Sun et al., 2018; He et al., 2021), we denote the mentioned entities in the given question q as the topic entity e_T, and assume it has been linked to some specific entity on the KG. Starting from the topic entity, we perform the invoking-linearization-generation procedure two times using the two interfaces in KG sequentially. First, we invoke the interface *Extract_Neighbor_Relation*(e_T) to extract the candidate one-hop relations, linearize them to compose the input prompt, and then leverage the LLM to select the useful relations $\{r\}$ according to the question. Then, based on $\{r\}$, we invoke the *Extract_Triples* (e_T, $\{r\}$) interface to collect the relevant triples for the head entity e_T and relation in $\{r\}$, then linearize these information, and finally employ the LLM to select the most relevant triples, whose tail entities will be considered as the final answer. Besides, we can also consider the multi-hop KGQA task (Lan et al., 2021), where after selecting the triples of the current hop, the LLM should assess whether the current information is sufficient to answer the question. Then, LLMs will make according actions based on the assessement, *i.e.,* stoping the iterations for producing the answer or continue the iterations on next-hop tail entities from selected triples.

Table-based Question Answering (TableQA). For TableQA, we typically need to answer the question according to the content in the given table. We also perform the above procedure by using the three interfaces in turn. Concretely, first, we invoke *Extract_Column_Name* ($\mathcal{T}$) to extract all column names of a table, linearize them, and leverage LLMs to select the relevant ones $\{c\}$ according to the question. Then, we invoke *Extract_Columns* ($\mathcal{T}$, $\{c\}$) to extract the contents of all relevant columns, and select the useful row indices $\{j\}$ by LLMs. Subsequently, we further invoke *Extract_SubTable* ($\mathcal{T}$, $\{c\}$, $\{j\}$) to generate the sub-table for the question. Based on the linearized sub-table, the LLM finally generates the answer to the question.

DB-based Semantic Parsing (Text-to-SQL). This task focuses on generating a SQL query that can be executed to obtain the required information from a database. To achieve this goal, first, we invoke *Extract_Table&Column_Name* ($\mathcal{D}$) to obtain all the table names and their column names in the DB, linearize them, and utilize the LLM to select the relevant table names. Then, we invoke *Extract_Tables_Information* ($\{\mathcal{T}\}$) to obtain all the relevant information (*i.e.,* column names and foreign keys) from these tables. Similarly, by linearizing these information and composing the input prompt, the LLM can generate an executable

[3]Note that our used prompts are not always consistent with the two examples, as we have rewritten them to better adapt into the specific datasets and extracted information.

SQL for the given question.

4 Experiment

To verify the effectiveness of our approach, we conduct experiments on three complex reasoning tasks over structured data, *i.e.,* KG based QA, Table based QA and DB based semantic parsing.

4.1 Experimental Settings

Here, we introduce the datasets, evaluation metrics, and baselines used in our experiment.

4.1.1 Datasets

For KG based QA (KGQA), we adopt two benchmark datasets, *i.e., WebQuestionsSP* (WebQSP) (Yih et al., 2016) and *MetaQA* (Zhang et al., 2018) for evaluation. The answer entities in WebQSP require up to 2-hop reasoning on the Freebase KG. In contrast, MetaQA contains questions in the movie domain, whose answer entities are up to 3 hops away from the topic entities on a movie KG (based on OMDb). According to the number of hops, it is split into three sub-datasets, *i.e.,* MetaQA-1hop, MetaQA-2hop, and MetaQA-3hop.

For Table based QA (TableQA), we adopt three widely-used datasets, *weakly-supervised WikiSQL* (WikiSQL) (Zhong et al., 2017), *WikiTableQuestions* (WTQ) (Pasupat and Liang, 2015), and *TabFact* (Chen et al., 2020). The first two ones are typical table-based question answering datasets, and the third one is a multiple-choice dataset that concentrates on table fact verification. WikiSQL requires filtering and aggregating information over the table content, and the WTQ demands more advanced reasoning capabilities (*e.g.,* sorting). TabFact needs to judge whether the provided statement agrees with the facts stored in a table.

For DB based semantic parsing (Text-to-SQL), we adopt three public datasets, *i.e., Spider* (Yu et al., 2018), *Spider-SYN* (Gan et al., 2021), and *Spider-Realistic* (Deng et al., 2021). Spider is a typical Text-to-SQL dataset covers 20 databases with a set of 1034 evaluation samples. Spider-SYN and Spider-Realistic are two more challenging datasets derived from Spider. Concretely, Spider-SYN manually substitutes the synonyms in natural language questions, while Spider-Realistic removes the questions in the evaluation set that explicitly mention the required columns name.

4.1.2 Evaluation Metrics

For KGQA, we employ Hits@1 that assesses whether the top-1 predicted answer is correct. In our approach, we focus on generating the most confident answer and then check if the prediction hits any target. As LLMs may generate multiple answers, we also conducted a manual double-check in the end (Tan et al., 2023), to judge if wrong answers are included. For TableQA, we adopt two evaluation metrics, namely denotation accuracy and accuracy. In WTQ and WikiSQL, denotation accuracy is employed to evaluate whether the predicted answer is same as the gold answer based on set-level equivalence. In TabFact, we adopt accuracy to assess the correctness of the prediction. For Text-to-SQL, we mainly adopt the execution accuracy (EX), which assesses whether the execution results of the predicted SQL and the gold SQL are the same.

4.1.3 Baselines

We compare our StructGPT with competitive full-data supervised-tuning baselines specially for these tasks. Specifically, the StructGPT is implemented as ChatGPT equipped with our invoking-linearization-generation procedure. For KGQA, we select KV-Mem (Miller et al., 2016), GragtNet (Sun et al., 2018), EmbedKGQA (Saxena et al., 2020), NSM (He et al., 2021), and UniKGQA (Jiang et al., 2022b). For TableQA, we select MAPO (Liang et al., 2018), TAPAS (Herzig et al., 2020; Eisenschlos et al., 2020), UnifiedSKG (T5-3B) (Xie et al., 2022), TAPEX (Liu et al., 2022), and DATER (Ye et al., 2023). For Text-to-SQL, we select RAT-SQL+BERT$_{Large}$ (Wang et al., 2020), TKK-Large (Gao et al., 2022a), T5-3B+PICARD (Raffel et al., 2020), RASAT+PICARD (Qi et al., 2022), and RESDSQL-3B+NatSQL (Li et al., 2023).

Besides, we also add a baseline method that directly uses ChatGPT to accomplish the above tasks in a zero-shot manner. To compare it with our approach, we utilize the same instructions in our approach to implement this method, to guarantee that the only difference is the usage of structured data. Specifically, in KGQA datasets, we follow existing work (Tan et al., 2023) that utilizes ChatGPT to answer the questions without using KG. In TableQA and Text-to-SQL, we feed the required information of tables with questions into ChatGPT (Liu et al., 2023c,a), without special treatment for the overlength problem.

Table 1: Performance comparison of different methods for KGQA (Hits@1 in percent). We copy the results in the first block from He et al. (2021) and (Jiang et al., 2022b). The best results of each block are highlighted in bold.

Methods	**WebQSP**	**MetaQA 1hop**	**MetaQA 2hop**	**MetaQA 3hop**
KV-Mem	46.7	96.2	82.7	48.9
GraftNet	66.4	97.0	94.8	77.7
EmbedKGQA	66.6	97.5	98.8	94.8
NSM	68.7	97.1	**99.9**	98.9
UniKGQA	**75.1**	**97.5**	99.0	**99.1**
ChatGPT	61.2	61.9	31.0	43.2
StructGPT	**72.6**	**94.2**	**93.9**	**80.2**

Table 2: Performance comparison of different methods for TableQA (denotation accuracy for WTQ and WikiSQL, accuracy for TabFact). We copy the results in the first block from their original papers, except for the result of TAPAS on TabFact, which is copied from Eisenschlos et al. (2020). The best results of each block are highlighted in bold.

Methods	**WTQ**	**WikiSQL**	**TabFact**
MAPO	43.8	72.6	-
TAPAS	48.8	83.6	81.0
UnifiedSKG (T5-3B)	49.3	86.0	83.7
TAPEX	57.5	**89.5**	84.2
DATER	**65.9**	-	**93.0**
ChatGPT	43.3	51.6	82.9
StructGPT	**48.4**	**54.4**	**87.1**

Table 3: Performance comparison of different methods for Text-to-SQL (execution accuracy in percent). We copy the results of RAT-SQL+BERT$_{Large}$ and TKK-Large from Deng et al. (2021) and Gao et al. (2022a), respectively. And we copy the results of the other three methods in the first block from Liu et al. (2023b). The best results of each block are highlighted in bold.

Methods	**Spider**	**Spider-SYN**	**Spider-Realistic**
RAT-SQL + BERT$_{\text{Large}}$	72.3	-	62.1
TKK-Large	73.2	60.5	64.4
T5-3B + PICARD	79.3	69.8	71.4
RASAT + PICARD	80.5	70.7	71.9
RESDSQL-3B + NatSQL	**84.1**	**76.9**	**81.9**
ChatGPT	70.1	58.6	63.4
StructGPT	**74.8**	**62.0**	**67.9**

4.2 Results and Analysis

We show the results on KGQA, TableQA, and Text-to-SQL tasks and analyze them respectively.

4.2.1 Evaluation on KGQA

Table 1 shows the results on KGQA datasets. First, ChatGPT can achieve performance comparable to the supervised learning model GraftNet (*i.e.,* 61.2 *v.s.* 66.4) on the WebQSP dataset, in a zero-shot setting without using KGs. It demonstrates that ChatGPT indeed grasps a certain amount of knowledge that can help it answer complex questions. However, on more difficult datasets that require multi-hop reasoning (*e.g.,* MetaQA-2hop and MetaQA-3hop), ChatGPT performs not well. It indicates that ChatGPT can not solely rely on its owned knowledge to answer difficult questions, and its augmentation with KGs is necessary. In contrast, when incorporating our proposed method to access KG, the performance of ChatGPT can be substantially improved, indicating the effectiveness of our proposed method for supporting LLM reasoning over KG. In our approach, we devise interfaces for structured data to efficiently read relevant information, and leverage ChatGPT to extract useful part and perform reasoning. We iterate the reading-then-reasoning procedure on devised interfaces sequentially, which can progressively capture more useful detailed evidence for finally obtaining the answer.

4.2.2 Evaluation on TableQA

Table 2 shows the results on three TableQA datasets. First, with the full table as the prompt, ChatGPT can also achieve comparable performance as full-data supervised-tuning methods, but performs not well on more difficult WikiSQL datasets. It also indicates that LLMs have the capability of understanding the knowledge within table data to some extent. Second, our proposed method can also improve the performance of LLMs a lot. It indicates the effectiveness of our proposed method in helping LLMs reasoning over Table. Our approach provides a more effective way for ChatGPT to iteratively access and utilize the relevant information from the table, which reduces the influence from irrelevant and redundant information from the table.

4.2.3 Evaluation on Text-to-SQL

Table 3 shows the results on DB-based datasets. First, with all the information from DB (table names, column names, and foreign keys) as the prompt, ChatGPT has the capability of directly generating suitable SQL query of the question, performing well on all three datasets. Whereas, the performance of ChatGPT is not better than competitive full-data supervised-tuning methods, showing the difficulty of this task. As our proposed method can extract relevant tables and columns, it also alleviates the influence from irrelevant information for ChatGPT to generate the SQL query. The consistent performance improvements over the three datasets also indicate the effectiveness of our proposed method.

4.3 Case Study

In order to better understand the whole process of our method, we select one representative example for each type of structured data and present them in Figure 2. Given the question, the interfaces of the structured data are sequentially invoked to iteratively extract more useful and detailed information. In each iteration, we perform the invoking-linearization-generation procedure.

Specifically, for KG, we first invoke the Extract_Neighbor_Relations function to extract the neighboring relations (*e.g., birthplace, residence and education*) of the topic entity "*Harper Lee*", then linearize them and compose the input prompt. In the prompt, we utilize the instruction (*i.e.,* provide only one relevant relation that's present in the candidate) to elicit the LLM to generate the most relevant relation, *i.e., education*. Based on the selected relation, we further invoke the Extract_Triples function to extract the triples with the relation to the topic entity. After linearization, another instruction (*i.e.,* you just need to provide only one answer entity), is adopted for guiding the LLM to generate the final answer, *i.e., Monroe County High School*.

For table, we first invoke the Extract_Column_Name function to extract the column names from the table for linearization, and then design the prompt (*i.e.,* which columns are most relevant to answering the question?) for the LLM to select the useful columns, *i.e., District and Incumbent*. Then, by using the Extract_Columns and Extract_SubTable functions and proper instructions, we elicit the LLM to select the useful row indices (*i.e.,* item 8) and finally generate the answer (*i.e.,* 19th).

For database, we also first invoke the Extract_Table&Column_Name to extract all the table names and column names, linearize them and utilize the instruction (*i.e.,* which tables do you need to complete the SQLite SQL query?) to prompt the LLM. Then, based on the selected tables (*i.e.,* Dogs and Breeds), we further invoke the Extract_Tables_Information function and prompt the LLM via an instruction (*i.e.,* complete sqlite SQL query only with no explanation) to generate the SQL for the question, which can be executed to obtain the final answer.

4.4 Zero-Shot Error Analysis

Although our proposed approach can significantly boost the zero-shot performance of LLMs, it still underperforms state-of-the-art full-data supervised-tuned methods. To systemically analyze the shortcomings of our approach, we first select three datasets (*i.e.,* WebQSP, WTQ, and Spider) with different types of structured data, and randomly sample 50 error cases from each dataset. Then, we manually examine these failures and classify them into five categories as:

- **Selection Error**: the relevant information has not been selected by the LLM.
- **Reasoning Error**: given the extracted relevant information, the LLM fails to generate the ground-truth answer or SQL.
- **Generation Format Error**: the generated answer is in an abnormal format that fails to be identified.
- **Hallucination**: the results generated by the LLM are inconsistent with the extracted information.
- **Other Errors**: other uncategorizable errors.

We show the statistics in Figure 3. First, for the three datasets, the distributions of occurring errors are different. In WikiSQL, the frequencies of generation format, selection and reasoning errors are relatively uniform. Whereas, in WebQSP, the selection error is the major error type (74%), since the KGQA task requires to select the most relevant one from thousands of relations, which is not an easy work. In Spider, reasoning error occurs more (62%), since the Text-to-SQL task requires LLMs to generate a SQL that can be executed to obtain the answer, which is also hard for LLMs.

According the error distributions, it is promising to refine the designs in our approach, to specially improve the performance for the major error cases on each dataset. Concretely, we can devise more high-quality prompts that elicit LLMs to carefully make decision when selection and reasoning on KGQA and Text-to-SQL tasks, respectively. Besides, we also consider to add more interfaces and iteration turns for decomposing the hard iteration into multiple simple ones, to simplify the complex

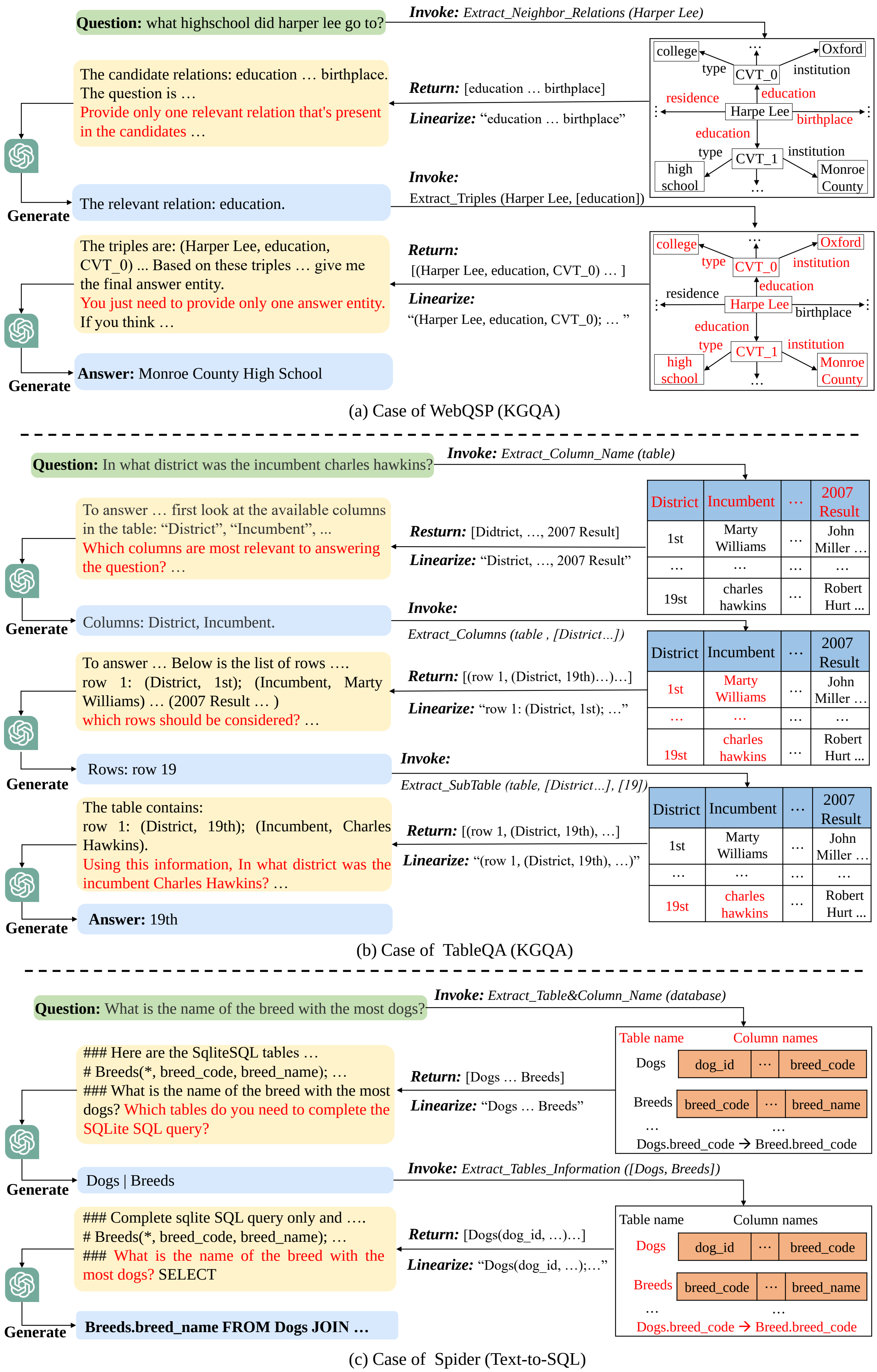

Figure 2: Case study of our method on KGQA, TableQA and Text-to-SQL task.

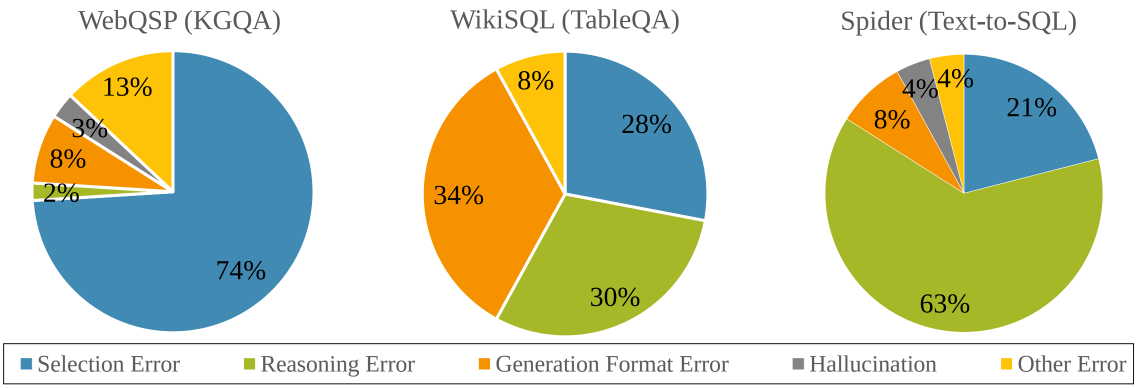

Figure 3: Proportions of different error types in three datasets over different types of structured data.

reasoning task for better performance. We will try the above solutions in our future work.

5 Conclusion

In this work, we proposed a general framework for improving the zero-shot reasoning ability of LLMs over structured data, namely StructGPT. In our approach, we first constructed the specialized interfaces that support accurate and efficient data access, and then proposed an invoking-linearization-generation procedure that leverages LLMs to read and perform reasoning based on the interface. By iterating the above procedure using the interfaces sequentially, LLMs can progressively capture more useful and detailed evidence and finally generate the answer. To verify the effectiveness of our approach, we implemented our approach on KG based QA, table based QA and DB based semantic parsing tasks. Experimental results on 8 datasets show that our approach can boost the zero-shot performance of LLMs in a large margin, and achieve comparable performance as full-data supervised-tuning methods. We also provide detailed case study and error analysis to point out the strengths and weakness of LLMs and our approach, for enlighting other researchers in related areas.

References

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *CoRR*.

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. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Yi Chen, Wei Wang, Ziyang Liu, and Xuemin Lin. 2009. Keyword search on structured and semi-structured data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 1005–1010. ACM.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-sql. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1337–1350.

Julian Martin Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, pages 281–296.

Denis Emelin, Daniele Bonadiman, Sawsan Alqahtani, Yi Zhang, and Saab Mansour. 2022. Injecting domain knowledge in language models for task-oriented dialogue systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11962–11974. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards robustness of text-to-sql models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2505–2515.

Chang Gao, Bowen Li, Wenxuan Zhang, Wai Lam, Binhua Li, Fei Huang, Luo Si, and Yongbin Li. 2022a. Towards generalizable and robust text-to-sql parsing. *arXiv preprint arXiv:2210.12674*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022b. PAL: program-aided language models. *CoRR*.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 553–561. ACM.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4320–4333.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *CoRR*, abs/2208.03299.

Jinhao Jiang, Kun Zhou, Ji-Rong Wen, and Xin Zhao. 2022a. $great truths are always simple: $ A rather simple knowledge encoder for enhancing the commonsense reasoning capacity of pre-trained models. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1730–1741. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2022b. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *CoRR*, abs/2212.00959.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4483–4491. ijcai.org.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Decoupling the skeleton parsing and schema linking for text-to-sql. *arXiv preprint arXiv:2302.05965*.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V. Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10015–10027.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023a. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *CoRR*, abs/2303.13547.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023b. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *CoRR*, abs/2303.13547.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Qian Liu, Fan Zhou, Zhengbao Jiang, Longxu Dou, and Min Lin. 2023c. From zero to hero: Examining the power of symbolic tasks in instruction tuning. *CoRR*, abs/2304.07995.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1400–1409.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022. SKILL: structured knowledge infusion for large language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1581–1588. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2022. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1535–1546. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *CoRR*, abs/2302.12813.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *arXiv preprint arXiv:2205.06983*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4498–4507.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4231–4242.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Evaluation of chatgpt as a question answering system for answering complex questions. *CoRR*, abs/2303.07992.

Yi Tay, Jason Wei, Hyung Won Chung, Vinh Q. Tran, David R. So, Siamak Shakeri, Xavier Garcia, Huaixiu Steven Zheng, Jinfeng Rao, Aakanksha Chowdhery, Denny Zhou, Donald Metzler, Slav Petrov, Neil Houlsby, Quoc V. Le, and Mostafa Dehghani. 2022. Transcending scaling laws with 0.1% extra compute. *CoRR*, abs/2210.11399.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578.

Xiaokai Wei, Shen Wang, Dejiao Zhang, Parminder Bhatia, and Andrew O. Arnold. 2021. Knowledge enhanced pretrained language models: A compreshensive survey. *CoRR*, abs/2110.08455.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 602–631.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning. *CoRR*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*.