Knowledge Base Question Answering: A Semantic Parsing Perspective

Yu Gu GU.826@OSU.EDU
Department of Computer Science and Engineering
The Ohio State University, USA

Vardaan Pahuja PAHUJA.9@OSU.EDU
Department of Computer Science and Engineering
The Ohio State University, USA

Gong Cheng GCHENG@NJU.EDU.CN
State Key Laboratory for Novel Software Technology
Nanjing University, China

Yu Su SU.809@OSU.EDU
Department of Computer Science and Engineering
The Ohio State University, USA

Abstract

Recent advances in deep learning have greatly propelled the research on semantic parsing. Improvement has since been made in many downstream tasks, including natural language interface to web APIs, text-to-SQL generation, among others. However, despite the close connection shared with these tasks, research on question answering over knowledge bases (KBQA) has comparatively been progressing slowly. We identify and attribute this to two unique challenges of KBQA, *schema-level complexity* and *fact-level complexity*. In this survey, we situate KBQA in the broader literature of semantic parsing and give a comprehensive account of how existing KBQA approaches attempt to address the unique challenges. Regardless of the unique challenges, we argue that we can still take much inspiration from the literature of semantic parsing, which has been overlooked by existing research on KBQA. Based on our discussion, we can better understand the bottleneck of current KBQA research and shed light on promising directions for KBQA to keep up with the literature of semantic parsing, particularly in the era of pre-trained language models.

1. Introduction

Modern knowledge bases (KBs) like FREEBASE [Bollacker et al., 2008] and WIKIDATA [Vrandečić and Krötzsch, 2014] store abundant world facts and organize them structurally following a crafted schema (i.e., an ontology). Knowledge base question answering (KBQA), which aims to locate the answers from the KB given a question in natural language, provides users with easy access to such massive structured data in KBs by offering a unified natural language interface to shield users from the heterogeneity underneath. State-of-the-art approaches to KBQA are predominantly based on semantic parsing, i.e., a question is mapped onto a logical form (e.g., SPARQL [Ravishankar et al., 2021] or λ-Calculus [Cai and Yates, 2013b]) that can be executed against the backend (i.e., a KB) to retrieve the answers (see Figure 1).

Neural models [Dong and Lapata, 2016, Jia and Liang, 2016, Hwang et al., 2019, Wang et al., 2020] have greatly enhanced the performance of semantic parsing, benefiting from recent advances such as the encoder-decoder framework [Sutskever et al., 2014, Bahdanau et al., 2014] and pre-trained language models (PLMs) [Devlin et al., 2019, Raffel et al., 2020]. However, these advances

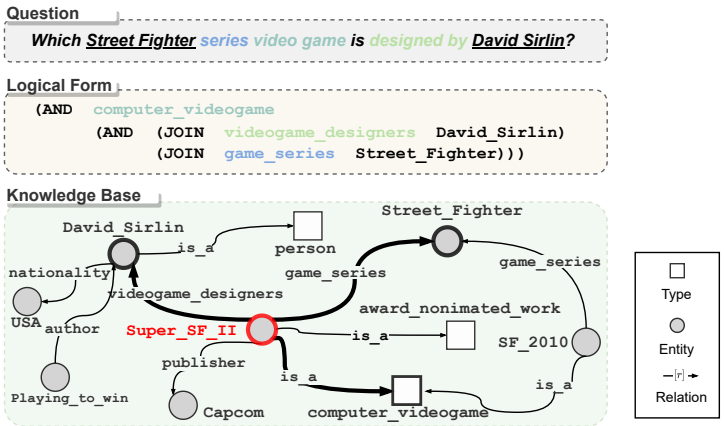

Figure 1: Most state-of-the-art approaches to KBQA are based on semantic parsing, i.e., a question is translated into a logical form (an S-expression [Gu et al., 2021] in this example), which is then executed over the KB to retrieve the answer (i.e., `Super_SF_II`). The semantic parsing task (implicitly) entails learning an alignment between the utterance and the KB schema (i.e., *schema linking*). We highlight the alignment using the same color. Named entities (marked with underlines) are usually linked beforehand to reduce the search space. Finally, the logical form composed from the linked items should be able to be grounded to the KB (i.e., *faithfulness*), where we highlight the grounding with bold lines.

have mainly inspired semantic parsing over a backend with a relatively simple schema (e.g., relational databases [Yu et al., 2018] and web APIs [Su et al., 2017]). They have been largely underexploited in KBQA due to several unique challenges facing large-scale KBs.

Semantic parsing in KBQA is uniquely challenging for two main reasons: *schema-level complexity* and *fact-level complexity*. On the one hand, the schema of a modern KB is extremely rich. For example, FREEBASE has over 8K schema items in total (6K relations and 2K types), while a relational database usually comprises dozens of schema items only (i.e., table names and column headers) [Yu et al., 2018]. Therefore, learning an alignment between natural language and the schema (i.e., schema linking) is much more challenging in KBQA. Also, the KB ontology models more complex relationships among schema items, (e.g., type hierarchy and domain/range information of relations). Therefore, a deeper understanding of the schema is in demand for KBQA compared with other semantic parsing tasks.

On the other hand, although content-agnostic models suffice to produce good results in other semantic parsing tasks, fact-level information (i.e., contents in the database) plays an integral role in KBQA. This is because the schema of a KB is instantiated dynamically. For instance, in FREEBASE's schema, the type `person` can be associated with over 1K different relations, while each instance of `person` may only be associated with a few of them (e.g., `David_Sirlin` in Figure 1 only has 4 relations). Consequently, generating logical forms that can ground to non-empty answers from the KB, i.e., *faithful* to the KB, requires tightly incorporating fact-level information. In addition, the graph structure of KB facts leads to an enormous search space due to combinatorial explosion, rendering generating faithful queries even more challenging. As a result, the encoder-decoder framework, which has been the *de facto* choice for many state-of-the-art semantic parsers, can be hardly adopted for KBQA.

We discuss the recent progress in KBQA from a semantic parsing perspective. Different from existing surveys which either endeavor to offer an inclusive overview of works on KBQA from dif-

Figure 2: We survey KBQA research based on semantic parsing and draw insights from neighboring tasks (e.g., text-to-SQL) in the broad literature of semantic parsing.

ferent communities [Diefenbach et al., 2018, Chakraborty et al., 2021], or take a task-oriented view and focus on a specific branch of KBQA (i.e., KBQA with complex questions [Fu et al., 2020, Lan et al., 2021a,b]), we focus on semantic parsing-based KBQA, the most prevailing KBQA methods, and discuss both challenges and solutions with our proposed taxonomy—therefore a technique-oriented view—while other methods (e.g., information retrieval based KBQA) are not our focus (see Figure 2). Notably, previous surveys only discuss existing works within the KBQA realm, while no effort has been made to situate KBQA in the broader literature of semantic parsing. Our survey fills this gap, which is critical for understanding the limits of current KBQA research and identifying future opportunities by taking inspiration from recent trends in semantic parsing.

2. Background

2.1 Knowledge Base

A knowledge base (KB) $\mathcal{K}$ stores facts in a structured way. It comprises two parts: an ontology $\mathcal{O}$ and a model $\mathcal{M}$. The ontology $\mathcal{O}$ defines rules about how facts should be organized in $\mathcal{K}$ (i.e., it defines the class hierarchy and domain/range information for relations). $\mathcal{M}$ instantiates $\mathcal{O}$ and represents facts, where each fact is a triple of the form *(subject, relation, object)*. Formally, $\mathcal{O} \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$ and $\mathcal{M} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{C} \cup \mathcal{E} \cup \mathcal{L})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of binary relations, $\mathcal{C}$ is a set of classes, and $\mathcal{L}$ is a set of literals, and $\mathcal{C} \cup \mathcal{R}$ constitute the schema items of $\mathcal{K}$. Compared with relational databases, KBs feature much more sophisticated schemas. Also, facts are typically organized into a graph rather than rows in a table. KBs thus encode more structured information.

2.2 KBQA Task

Task formulation. Given $\mathcal{K} = \mathcal{O} \cup \mathcal{M}$ and a question q in natural language, typically, the task of KBQA is to find a set of entities $\mathcal{A} \subseteq \mathcal{E}$, with $|\mathcal{A}| \geq 1$, that capture the intent of q. $\mathcal{A}$ can be directly returned as the answer or further aggregated (e.g., applying a counting function). For example, given a question *"Which street Fighter series video game is designed by David Sirlin?"* the goal is to find the answer entity `Super_SF_II`, with $\mathcal{K}$ being FREEBASE (see Figure 1).

Logical forms. Logical forms (i.e., queries) for KBQA can be in different meaning representations (MRs), e.g., SPARQL [Yih et al., 2016], graph query [Su et al., 2016], λ-DCS [Cai and Yates, 2013b], and S-expression [Gu et al., 2021]. λ-DCS and S-expression can be viewed as the linearized version of graph query and better suits encoder-decoder models. Also, they can be viewed as the syntactic sugar for SPARQL.[1] An example of a query in S-expression is depicted in Figure 1. A *well-formed* query is syntactically correct and thus can be executed with no exception, but possibly with an empty answer, while a *faithful* query must find non-empty answers from the KB.

1. This claim does not hold generally because the different expressiveness of MRs, but holds in our context because KBQA usually only exploit a subset of expressiveness for most MRs.

Dataset	KB	Size	Logical Form	Generalization Assumption
QALD [Ngomo, 2018]	DBPEDIA	558	SPARQL	comp.
LC-QUAD [Trivedi et al., 2017]	DBPEDIA	5,000	SPARQL	i.i.d.
LC-QUAD 2.0 [Dubey et al., 2019]	WIKIDATA, DBPEDIA	30,000	SPARQL	i.i.d.
CSQA [Saha et al., 2018]	WIKIDATA	800,000	N/A	i.i.d.
KQA PRO [Shi et al., 2020]	WIKIDATA	117,970	Program	i.i.d.
FREEBASE917 [Cai and Yates, 2013a]	FREEBASE	917	λ-Calculus	zero-shot
WEBQ [Berant et al., 2013]	FREEBASE	5,810	N/A	i.i.d.
WEBQSP [Yih et al., 2016]	FREEBASE	4,737	SPARQL	i.i.d.
COMPLEXWEBQ [Talmor and Berant, 2018]	FREEBASE	34,689	SPARQL	i.i.d.
GRAPHQ [Su et al., 2016]	FREEBASE	5,166	Graph query	comp.+zero-shot
GRAILQA [Gu et al., 2021]	FREEBASE	64,331	S-expression	i.i.d.+comp.+zero-shot

Table 1: Information on existing KBQA datasets. For the generalization assumption, we follow the definitions by Gu et al. [2021]

Categories	Sub-tasks	Solutions
Ranking	*Candidate Enumeration*	Template-based: Bast and Haussmann [2015], Berant and Liang [2014], Abujabal et al. [2017]; KB traversal: Yih et al. [2015], Zafar et al. [2018], Hu et al. [2018], Lan et al. [2019a], Gu et al. [2021]$^{\heartsuit}$, Ye et al. [2021]$^{\heartsuit}$
	Semantic Matching	Learning-to-rank: Bast and Haussmann [2015], Abujabal et al. [2017], Yih et al. [2015], Zafar et al. [2018], Hu et al. [2018], Lan et al. [2019a]; PLMs: Gu et al. [2021]$^{\heartsuit}$, Ye et al. [2021]$^{\heartsuit}$; Paraphrasing: Berant and Liang [2014]
Coarse-to-fine	*Skeleton Parsing*	Question decomposition: Ding et al. [2019], Bhutani et al. [2019], Hu et al. [2021]$^{\heartsuit}$; Encoder-decoder: Ravishankar et al. [2021]$^{\heartsuit}$, Das et al. [2021]$^{\heartsuit}$, Zhang et al. [2019]; Pipeline:Sun et al. [2020]$^{\heartsuit}$; AMR parsing: Kapanipathi et al. [2021], Bornea et al. [2021]$^{\heartsuit}$
	Grounding	Filling partial results: Ding et al. [2019], Bhutani et al. [2019], Hu et al. [2021]$^{\heartsuit}$; Refining: Ravishankar et al. ; [2021]$^{\heartsuit}$, Das et al. [2021]$^{\heartsuit}$; Using training data: Sun et al. [2020]$^{\heartsuit}$ Tranpiling: Kapanipathi et al. [2021], Bornea et al. [2021]$^{\heartsuit}$
Generation	-	Graph search: Lan et al. [2019b], Chen et al. [2019], Lan and Jiang [2020]$^{\heartsuit}$; Unconstrained decoding: Zhang et al. [2019], Yin et al. [2021], Gu et al. [2021]$^{\heartsuit}$, Banerjee et al. [2022]$^{\heartsuit}$; Schema-level constrained decoding: Chen et al. [2021]$^{\heartsuit}$, Cao et al. [2021]$^{\heartsuit}$; Fact-level constrained decoding: Liang et al. [2017], Ansari et al. [2019], Qiu et al. [2020], Gu and Su [2022]$^{\heartsuit}$

Table 2: We summarize representative KBQA studies based on semantic parsing. Existing works fall into three families: *ranking*, *coarse-to-fine*, and *generation* methods. Methods using PLMs are indicated with $^{\heartsuit}$.

Evaluation. To benchmark the research on KBQA, a line of datasets have been released during the past decade (summarized in Table 1). Questions in all datasets except for CSQA, which only contains machine-generated questions, are either collected from web search logs (i.e., WEBQ) or paraphrased by crowd workers. Most datasets are typically collected with synthetic queries with pre-defined templates or random sampling. Different datasets may serve different purposes. For example, COMPLEXWEBQ is a dataset that particularly focuses on complex questions. GRAILQA is a dataset that can systematically evaluate the generalizability of KBQA models beyond i.i.d. level.

Generally, there are two types of metrics for evaluation: execution-based metrics (e.g., $\boldsymbol{F}_1$), which encourage partially correct prediction, and more strict metrics based on logical form (e.g., **ExactMatch**).

3. Approaches

A potpourri of approaches have been proposed to address the uniquely challenging task of KBQA. We categorize them into three families: *ranking methods*, *coarse-to-fine methods*, and *generation methods* (Figure 3). Despite the disparate designs, all methods share the same overarching idea, i.e., using the KB structure (either schema-level or fact-level) to constrain output space. Specifically, ranking methods first enumerate candidate queries from the KB and semantic parsing thus boils

down to computing the matching score for each candidate-question pair. Coarse-to-fine methods first generate query skeletons and then ground the skeletons to the KB with admissible schema items. Generation methods, which have emerged more recently, typically ground a query to the KB via constrained decoding, and thus dynamically reduce the search space. These methods all leverage the KB structure to reduce the search space for schema linking, hence handling the *schema-level complexity*. Also, they address the *fact-level complexity* and produce faithful queries using pre-processing (i.e., candidate enumeration), post-processing (i.e., skeleton grounding), and online processing respectively (i.e., constrained decoding). Tabel 2 summarizes existing works in different families.

3.1 Ranking Methods

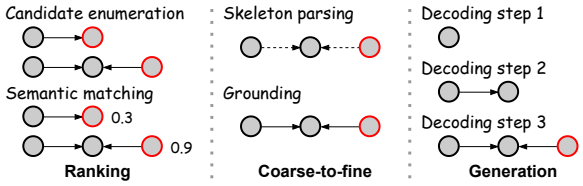

Figure 3: Illustrations of the high-level ideas of three semantic parsing-based KBQA families. Each query is represented as a graph (here a path for brevity). Entities or query nodes are both represented as a circle, where the red circle denotes the target node. An edge denotes a KB relation while a dashed edge denotes a relation placeholder to be grounded.

Ranking methods decompose KBQA into two disjointed sub-tasks, i.e., *candidate enumeration* and *semantic matching*. Candidate enumeration incorporates fact-level information by directly listing faithful queries from the KB. Semantic matching aims at returning a matching score for each question-candidate pair and is usually modeled as a machine learning task.

Candidate Enumeration. Traditional methods pre-specify a set of query templates and then fill the templates with concrete arguments. For example, Bast and Haussmann [2015] and Berant and Liang [2014] only define three and five query templates respectively and fill the template slots for entities or schema items using heuristics. Abujabal et al. [2017] mine query templates from training data and do slot filling with a lexicon. Another common strategy is to first identify a topic entity from the question and then use it as an anchor to traverse its neighborhood in the KB to enumerate possible queries. For example, Yih et al. [2015] first enumerate core inferential chains (i.e., relation paths) starting from the topic entity and then extend the chains by adding possible constraints or aggregation functions to form candidate queries. Hu et al. [2018] extend their work to enumerate more diverse candidate queries using crafted operations over the KB (e.g., merge and expand). Similarly, Gu et al. [2021] and Ye et al. [2021] also enumerate relation paths starting from identified entities. Lan et al. [2019a] seek to use more anchors other than named entities, such as entities of common nouns and relation types, to guide the candidate enumeration, while Zafar et al. [2018] also use a set of candidate relations to generate candidate queries. Though these methods can effectively use identified anchors for faithful query enumeration, they all suffer from scalability, e.g., they typically impose a maximum length limit of 2 for relation paths.

Semantic Matching. Early methods rely on hand-crafted features and learning-to-rank approaches to find the top-ranked candidate [Bast and Haussmann, 2015, Yih et al., 2015, Abujabal et al., 2017, Hu et al., 2018, Zafar et al., 2018, Lan et al., 2019a]. Commonly used features include simple statistics such as the number of grounded entities and features retrieved by neural models, e.g., Yih et al. [2015] use a CNN model to compute a matching score for each pair of question and core relation,

while a Tree-LSTM [Tai et al., 2015] is used by Zafar et al. [2018]. Recent methods start to use PLMs for semantic matching. Gu et al. [2021] first train a BERT-based Seq2Seq model and then use it as a scorer, while Ye et al. [2021] directly use BERT to get the matching score by linearizing each candidate query and formulating each question-candidate pair as a sentence pair input to BERT. Berant and Liang [2014] pursue a different direction by converting candidate queries into canonical utterances and modeling the semantic matching task as paraphrasing.

3.2 Coarse-to-fine Methods

Coarse-to-fine methods decompose semantic parsing into two stages. First, a model only predicts a rough query skeleton (or a sketch), which focuses on the high-level structure (*a coarse parse*). Second, the model fills the missing details by grounding the query skeleton to the KB (*a fine parse*). The intuition is to disentangle information at different levels of granularity in semantic parsing [Dong and Lapata, 2018, Zhang et al., 2019]. In addition, the two-stage searching more effectively prunes the search space resulting from combinatorial explosion in KBQA, compared with ranking methods.

Skeleton Parsing. Ding et al. [2019] learn frequent query substructures from training data and generate a query skeleton that merges them. Hu et al. [2021] rely on manually-crafted rules operating over constituency parses to generate a special query skeleton, named entity description graph . Bhutani et al. [2019] predict a computation plan for the question, which indicates how a complex question should be decomposed into several sub-questions. Different from them, Sun et al. [2020] perform skeleton parsing in a pipeline with sub-tasks like question split and span prediction. Recent works predict a skeleton from the question using encoder-decoders. Ravishankar et al. [2021] decode a sketch for SPARQL with textualized relation placeholders that can support cross-KB generalization. Bornea et al. [2021], Kapanipathi et al. [2021] delegate skeleton parsing to an AMR parser pre-trained on external corpora. Das et al. [2021] use T5 [Raffel et al., 2020] to directly output a coarse query from the question and retrieved cases from training data. The query may contain inaccurate schema items that need further revision during grounding.

Grounding. The grounding step fills (or revises) a query skeleton to produce the final faithful query. Ding et al. [2019] use off-the-shelf systems to link possible entities and relations in the question, and then try all possible combinations of them for each skeleton. The final score is based on both the linking probability and skeleton parsing probability. Hu et al. [2021] ground the skeleton by mapping entity attributes in natural language into KB relations using BERT as a binary classifier. Bhutani et al. [2019] ground their computation plan by predicting each partial query using a semantic matching module based on LSTM and word embeddings. Sun et al. [2020] find the most similar question from training data and use the schema items in that question to ground the skeleton. Ravishankar et al. [2021] rely on a BERT-based encoder to map the KB-agnostic relation placeholders onto a concrete relation in the target KB. Bornea et al. [2021], Kapanipathi et al. [2021] transpile AMR into SPARQL queries with specified mapping rules and neural modules for relation linking and decoding. Das et al. [2021] revise the inaccurate schema items by aligning them with items in the neighborhood of the topic entity using both string-level and embedding-level similarities.

3.3 Generation Methods

Generation methods have been the de facto choice for many semantic parsing tasks and have also been a trending paradigm for KBQA due to their flexibility and scalability. Adapting generation

methods to KBQA poses a unique challenge for producing faithful queries, which requires tightly incorporating the KB structure during generation (decoding).

Graph Search Paradigm. Several works capitalize on the intuition that a faithful query can be produced with direct graph search over the KB given anchor entities. Lan et al. [2019b], Chen et al. [2019] propose to perform beam search over the KB to find the top-K relation paths, i.e., at each step, they rank the reachable relations given the context and extend the beam paths with the top-ranked ones. Lan and Jiang [2020] extend this line of work by introducing more operations for graph search. In addition to extending the paths, they define two more actions: *connect* and *aggregate*, where connect adds an extra constraint to a partial query and aggregate assigns an aggregation function over variables. A key question for these methods is how to condition the search process on the input question. Also, the conditioning should be dynamic during the search process. Chen et al. [2019] propose a dynamic question representation module which generates a new question representation at each search step. Lan and Jiang [2020] use BERT as a sentence-pair classifier to provide a matching score feature between each partial query and the question at each step.

Encoder-Decoder Paradigm. Encoder-decoders have actually offered standard solutions not only to conditioning the search on the input question via attention mechanism [Bahdanau et al., 2014, Vaswani et al., 2017] but also to modeling decoding history and termination. Yin et al. [2021] directly applies an encoder-decoder framework to translate a question into a SPARQL query, where SPARQL's syntactic symbols are preprocessed to ease the learning. Banerjee et al. [2022] applies more advanced pre-trained encoder-decoders like T5 and BART [Lewis et al., 2020], assuming entities and relations in the target query are given as input. However, both methods adopt an unconstrained decoder which predicts free-formed queries from the enormous search space with no faithfulness guarantee. Gu et al. [2021] propose to build a question-specific decoding vocabulary, where only schema items reachable from the identified entities within 2 hops are included. Their method can significantly reduce the search space but has guarantees for neither well-formedness nor faithfulness. Chen et al. [2021] and Cao et al. [2021] impose schema-level constraints during decoding to prune the search space with a grammar-based decoder and a function-based decoder respectively. Queries predicted by them are well-formed but may still be unfaithful. To provide a faithfulness guarantee, a more effective solution is to impose fact-level constraints for prediction. Specifically, Liang et al. [2017] propose to predict a query token by token, where a set of admissible tokens[2] is derived based on the decoding history and intermediate executions at each step. Ansari et al. [2019], Qiu et al. [2020] share the same spirit but instead of operating over the tokens space, they predict an action (e.g., choosing a function or an argument) at each step to better suit semantic parsing. Gu and Su [2022] also adopt token-based constrained decoding, while they propose a novel contextualized encoder fueled by PLMs and their decoder can support more types of KB queries.

3.4 Training

KBQA models are trained with either *strong supervision* (i.e., question-query pairs) or *weak supervision* (i.e., question-answer pairs). The choice of supervision can be orthogonal to the model architecture. Training from weak supervision demands effectively searching for a set of proxy target queries, after which the optimization process resembles learning with strong supervision, with the

2. A token is admissible if it can lead to a faithful query. Their entire token vocabulary comprises schema items, intermediate variables, and syntactic items like function names or brackets.

difference being that each proxy query is weighed based on the similarity between its execution and the gold answers [Yih et al., 2015]. Searching for proxy queries can be modeled with reinforcement learning, where the weights define the rewards. One major challenge arises from the sparse reward signal at the early stages. To address it, different techniques, such as pre-training [Qiu et al., 2020] and iterative ML training [Liang et al., 2017], have been proposed to warm up the model. Note that which type of supervision is more advantageous remains to be further investigated. Yih et al. [2016] point out that training with weak supervision may yield sub-optimal results due to spurious proxy queries, while Qin et al. [2020] find that using multiple proxy queries provides more comprehensive information and leads to better performance.

3.5 Empirical Results

In Table 3, we show the F_1 and the corresponding family of best-performing models on KBQA benchmarks that have at least two published semantic parsing-based models evaluated. On KQA PRO, WEBQSP, and GRAPHQ, the state-of-the-art models are based on generation [Lewis et al., 2020, Gu and Su, 2022], while ranking methods achieve the best results on LC-QUAD and GRAILQA [Zafar et al., 2018, Ye et al., 2021] and the best performance on COMPLEXWEBQ is obtained by the coarse-to-fine method [Das et al., 2021]. Though no family dominates all the benchmarks, generation methods tend to be a trending option due to the easy integration of PLMs—all best-performing models are based on PLMs except for LC-QUAD.[3] We also want to note that the results on KQA PRO are relatively higher and the results on GRAPHQ are relatively lower because KQA PRO uses a down-sampled KB which leads to smaller search space while GRAPHQ only evaluates on challenging non-i.i.d. questions with a small training set of 2583 training questions. For more empirical results, we refer readers to Perevalov et al. [2022].

Dataset	Top-1 F_1	Top-1 Family
LC-QUAD	75.0[Zafar et al., 2018]	Ranking
KQA PRO♣	90.6[Lewis et al., 2020]♡	Generation
WEBQSP	75.3[Gu and Su, 2022]♡	Generation
COMPLEXWEBQ♣	70.0[Das et al., 2021]♡	Coarse-to-fine
GRAPHQ	31.8[Gu and Su, 2022]♡	Generation
GRAILQA♣	74.4[Ye et al., 2021]♡	Ranking

Table 3: We present the best-published results on KBQA benchmarks with at least two semantic parsing-based models evaluated on them. ♣ indicates benchmarks with official evaluation scripts. ♡ indicates using PLMs.

4. Semantic Parsing Literature

We briefly review research on semantic parsing, focusing on several recent trends. By drawing insights from trends in the broader literature of semantic parsing, we can better understand the bottleneck of current KBQA research and discuss promising future directions (Section 5).

From Pipeline to End-to-End. Traditional semantic parsing methods are typically based on pipelines. For example, early works first build a lexicon of phrases paired with meaning representations. Then, given an input utterance, they identify relevant lexical entries from the lexicon and apply combination rules to synthesize the logical form in a bottom-up manner [Zettlemoyer and Collins, 2005, Cai and Yates, 2013b, Berant et al., 2013]. Both constructing a lexicon and specifying combination rules require domain knowledge and thus suffer from flexibility, e.g., Berant et al. [2013] use domain-specific corpora like ClueWeb to construct their lexicon, while Zettlemoyer and Collins [2005] and Cai and Yates [2013b] need to define a CCG for combination rules. To provide a

3. There is no published result on LC-QUAD using PLMs so far.

more general solution, Dong and Lapata [2016] adapt the Seq2Seq model to address semantic parsing by modeling it as a sequence transduction task. The end-to-end encoder-decoder paradigm has since been the de facto choice for many semantic parsing tasks. However, a vanilla Seq2Seq model which predicts free-formed sequences can be sub-optimal for semantic parsing. To better adapt to the encoder-decoder paradigm for semantic parsing, a common strategy is to adopt a grammar-based decoder which only outputs well-formed queries [Krishnamurthy et al., 2017, Wang et al., 2020]. More recently, Rubin and Berant [2021] propose a novel semi-autoregressive bottom-up decoder that achieves better efficiency with parallelization.

Semantic Parsing with Pre-Training. PLMs are versatile for a wide range of NLP tasks due to their general knowledge of language. Unsurprisingly, they have also received much success in semantic parsing. Encoder-decoder-based semantic parsers usually use PLMs to provide better contextualized representations for both the question and the schema of the target backend. Specifically, a common practice is to concatenate the question and all schema items together as the input to the PLM, and thus contextualization is achieved via the PLM's self-attention layers [Hwang et al., 2019]. Recent works have also used PLMs for decoding. Encoder-decoder PLMs are trained with unstructured textual data and have an unconstrained output space, which is different from semantic parsing. To deal with it, PICARD [Scholak et al., 2021] finds well-formed output for text-to-SQL using PLMs by rejecting inadmissible tokens at each decoding step. Specifically, two different levels of checking are applied. First, lexical-level checking helps to reject invalid tokens from the output space (e.g., a misspelled column header). Second, schema-level checking helps to reject things like selecting a column from the table to which it does not belong.

In addition, continuing pre-training PLMs with an in-domain corpus (i.e., task-specific pre-training) can better tailor PLMs for semantic parsing [Yu et al., 2020, Herzig et al., 2020, Deng et al., 2021, Liu et al., 2021]. For example, GRAPPA [Yu et al., 2020] synthesizes an in-domain corpus of utterance-SQL pairs generated from a synchronous context-free grammar and proposes a binary classification task for pre-training called SQL semantic prediction (SSP), which predicts whether a table column appears in the target SQL query. With the task-specific corpus and objective, GRAPPA considerably improves the performance on text-to-SQL over general-purpose PLMs.

Out-of-Distribution Neural Semantic Parsing. Different from traditional semantic parsing methods [Zettlemoyer and Collins, 2005, Cai and Yates, 2013b], neural semantic parsers usually hold an i.i.d. assumption [Dong and Lapata, 2016, Hwang et al., 2019]. Semantic parsers that operate with an i.i.d. assumption may fail in real-life scenarios where true user distribution is hard to capture. In addition, training semantic parsers with weak generalizability can be data-inefficient. To support the study of out-of-distribution generalization for semantic parsing, several benchmarks have been released. Particularly, Yu et al. [2018] release Spider to evaluate cross-domain generalization in text-to-SQL. Each question in Spider is given its own target database, and databases in the test set are never seen during training. In this setting, models are encouraged to really perform semantic parsing rather than pattern memorization, i.e., the target query for a question may be seen during training under i.i.d. setting, which offers semantic parsers a shortcut to make predictions.

5. Discussion

To conclude this survey, we provide in-depth discussions on promising directions in KBQA, tightly drawing insights from the semantic parsing research discussed in Section 4.

5.1 Towards End-to-End KBQA

KBQA Based on Encoder-Decoders Despite the fact that the encoder-decoder paradigm has revolutionized the research in semantic parsing and become a norm for many downstream tasks, it is not as popular in KBQA. This is due to the challenge in generating faithful queries for KBQA using encoder-decoder models, as we discussed earlier in Section 3.3. Here we elaborate more on the difficulty in generating faithful queries for KBQA and text-to-SQL using encoder-decoder models. Specifically, encoder-decoder models can make predictions with three different levels of control, namely, unconstrained decoding, decoding with schema-level constraints, and decoding with fact-level constraints. For text-to-SQL, unconstrained decoding can achieve satisfactory results under the i.i.d. setting [Hwang et al., 2019], while for KBQA, a Seq2Seq model with unconstrained decoding considerably underperforms the ranking methods [Gu et al., 2021]. This is because KBs feature a much more sophisticated schema whose structures cannot be well learned in a data-driven manner using an unconstrained decoder. We can inject prior knowledge on the schema to constrain the decoding output space. For example, in text-to-SQL such constraints include selecting a column from the table it belongs to, while in KBQA such constraints can be defined based on a relation's domain/range information. Such schema-level constraints guarantee the well-formedness of both text-to-SQL and KBQA and can be easily implemented with grammar-based decoders [Krishnamurthy et al., 2017]. For existing text-to-SQL benchmarks, well-formedness almost equals to faithfulness, while this does not hold for KBQA because a KB is instantiated dynamically, e.g., not every `Person` is associated with the relation `videogame_designers`. As a result, more complicated fact-level constraints are indispensable for using encoder-decoders in KBQA.

Joint Entity Linking Existing studies on KBQA typically rely on off-the-shelf entity linkers built upon simple techniques like fuzzy string matching to identify topic entities during pre-processing. We argue that the current norm is questionable because 1) pipelines suffer from error propagation and 2) it prohibits entity linking and semantic parsing from boosting each other. Preliminary efforts have been made to break the norm via relation-enhanced entity disambiguation [Ye et al., 2021] and joint linking over relations and entities (but limited to a set of candidates filtered in advance). We envision a KBQA model that jointly performs entity linking and semantic parsing in a truly end-to-end manner. A promising direction is to project entities into a continuous space and perform differentiable operations over them [Ren et al., 2021] because operating over millions of entities in discrete space can be intractable.

5.2 Towards KBQA with Pre-Training

KBQA with PLMs The high volume of KB schema items prohibits KBQA from jointly encoding the question and all schema items using PLMs via input concatenation, as done in text-to-SQL. To adapt it to KBQA, Gu et al. [2021] and Chen et al. [2021] propose to narrow down the size of candidate schema items to fit PLMs' length limit. However, identifying relevant schema items beforehand is inflexible. A more promising direction is to integrate PLMs with constrained decoding [Liang et al., 2017, Gu and Su, 2022], where relevant schema items are selected on the fly. In addition, to better understand the KB schema, future works may explicitly model the relationships among schema items and question tokens [Wang et al., 2020], instead of only delegating it implicitly to PLMs' self-attention layers via input concatenation. For encoder-decoder PLMs, Xie et al. [2022] have shown that directly fine-tuning T5 suffices to outperform the prior art on COMPLEXWEBQ.

Their preliminary studies indicate the great potential of using encoder-decoder PLMs as a unified solution to semantic parsing. Augmenting the encoder-decoder PLMs with constrained decoding algorithms like PICARD [Scholak et al., 2021] is a promising direction to pursue in KBQA.

KBQA-specific Pre-training Existing works in KBQA have shown the feasibility of cross-dataset pre-training [Gu et al., 2021, Cao et al., 2021], while a general solution towards KBQA-specific pre-training like GRAPPA for text-to-SQL remains absent. Applying task-specific pre-training in KBQA is challenging. First, synthesizing an aligned corpus of question-query pairs may lead to data contamination, especially for evaluating zero-shot generalization in KBQA. Second, the KBQA-specific pre-training task needs to take account of the structured information in queries, while binary classification tasks like SSP will fall short of this goal. A possible direction to address them is to synthesize KB-agnostic queries and specify a structured prediction task for pre-training.

5.3 Towards More Generalizable KBQA

Similarly, research on KBQA has recently shifted its focus to non-i.i.d. generalization. Gu et al. [2021] release GRAILQA to systematically investigate the generalizability on three levels: *i.i.d.*, *compositional*, and *zero-shot*. The key difference between non-i.i.d. generalization in text-to-SQL and KBQA is that the relevant tables are given as input in text-to-SQL, while all questions share the same KB in KBQA, so the model needs to determine which part of the KB is relevant by itself. This is extremely challenging for a non-i.i.d. setting because the model can easily overfit the KB segments seen during training. Thus, Gu et al. [2021] suggest that highly generalizable models should feature effective search space pruning. A promising direction is to use a more KB-specific constrained decoding algorithm for encoder-decoder PLMs rather than PICARD [Scholak et al., 2021, Xie et al., 2022].

Though the definitions in Gu et al. [2021] are extendable to KBQA with multiple KBs, their dataset is collected with a single KB (i.e., FREEBASE). Recent works have addressed another type of non-i.i.d. generalization, i.e., evaluating on a KB different from training [Ravishankar et al., 2021, Cao et al., 2021]. Developing KBQA models with such cross-KB generalizability is a stepping stone towards the ambitious goal of question answering over the linked open data (LOD) cloud [Soru et al., 2020]. A dataset that supports systematic evaluation for such cross-KB generalization is in demand.

5.4 Other Trends

Several other trends in semantic parsing may also inspire research in KBQA. First, the human-in-the-loop methodology can effectively improve the accuracy of semantic parsing on complex questions that are challenging to solve in one shot [Gur et al., 2018, Yao et al., 2019]. Future works may study interactive KBQA to better handle complicated KB queries[Mo et al., 2022]. Second, prompting [Brown et al., 2020] has been successfully applied to semantic parsing [Schucher et al., 2022, Yang et al., 2022], while techniques like prompt tuning [Lester et al., 2021] in KBQA remain to be investigated. Related to this point, recent works have considered a few-shot setting for cross-domain generalization in text-to-SQL [Lee et al., 2021], future works may also explore the potential of few-shot in-context learning and prompting in non-i.i.d. generalization in KBQA.

References

A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum. Automated template generation for question answering over knowledge graphs. In *WWW*, 2017.

G. Ahmed Ansari, A. Saha, V. Kumar, M. Bhambhani, K. Sankaranarayanan, and S. Chakrabarti. Neural program induction for kbqa without gold programs or query annotations. In *IJCAI*, 2019.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv*, 2014.

D. Banerjee, P. Nair, J. Kaur, R. Usbeck, and C. Biemann. Modern baselines for sparql semantic parsing. In *SIGIR*, 2022.

H. Bast and E. Haussmann. More accurate question answering on freebase. In *CIKM*, 2015.

J. Berant and P. Liang. Semantic parsing via paraphrasing. In *ACL*, 2014.

J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, 2013.

N. Bhutani, X. Zheng, and H. Jagadish. Learning to answer complex questions over knowledge bases with query composition. In *CIKM*, 2019.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

M. Bornea, R. Astudillo, T. Naseem, N. Mihindukulasooriya, I. Abdelaziz, P. Kapanipathi, R. Florian, and S. Roukos. Learning to transpile amr into sparql. *arXiv*, 2021.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Q. Cai and A. Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, 2013a.

Q. Cai and A. Yates. Semantic parsing freebase: Towards open-domain semantic parsing. In *SEM*, 2013b.

S. Cao, J. Shi, Z. Yao, L. Hou, J. Li, and J. Xiao. Program transfer and ontology awareness for semantic parsing in kbqa. *arXiv*, 2021.

N. Chakraborty, D. Lukovnikov, G. Maheshwari, P. Trivedi, J. Lehmann, and A. Fischer. Introduction to neural network-based question answering over knowledge graphs. *Data Mining and Knowledge Discovery*, 2021.

S. Chen, Q. Liu, Z. Yu, C. Lin, J. Lou, and F. Jiang. ReTraCk: A flexible and efficient framework for knowledge base question answering. In *ACL Demo Track*, 2021.

Z. Chen, C. Chang, Y. Chen, J. Nayak, and L. Ku. UHop: An unrestricted-hop relation extraction framework for knowledge-based question answering. In *NAACL*, 2019.

R. Das, M. Zaheer, D. Thai, A. Godbole, E. Perez, Y. Lee, L. Tan, L. Polymenakos, and A. McCallum. Case-based reasoning for natural language queries over knowledge bases. In *EMNLP*, 2021.

X. Deng, A. Awadallah, C. Meek, O. Polozov, H. Sun, and M. Richardson. Structure-grounded pretraining for text-to-SQL. In *NAACL*, 2021.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

D. Diefenbach, V. Lopez, K. Singh, and P. Maret. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information systems*, 2018.

J. Ding, W. Hu, Q. Xu, and Y. Qu. Leveraging frequent query substructures to generate formal queries for complex question answering. In *EMNLP-IJCNLP*, 2019.

L. Dong and M. Lapata. Language to logical form with neural attention. In *ACL*, 2016.

L. Dong and M. Lapata. Coarse-to-fine decoding for neural semantic parsing. In *ACL*, 2018.

M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *ISWC*, 2019.

B. Fu, Y. Qiu, C. Tang, Y. Li, H. Yu, and J. Sun. A survey on complex question answering over knowledge base: Recent advances and challenges. *arXiv*, 2020.

Y. Gu and Y. Su. Arcaneqa: Dynamic program induction and contextualized encoding for knowledge base question answering. *arXiv*, 2022.

Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su. Beyond iid: three levels of generalization for question answering on knowledge bases. In *TheWebConf*, 2021.

I. Gur, S. Yavuz, Y. Su, and X. Yan. Dialsql: Dialogue based structured query generation. In *ACL*, 2018.

J. Herzig, P. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In *ACL*, 2020.

S. Hu, L. Zou, and X. Zhang. A state-transition framework to answer complex questions over knowledge base. In *EMNLP*, 2018.

X. Hu, Y. Shu, X. Huang, and Y. Qu. Edg-based question decomposition for complex question answering over knowledge bases. In *ISWC*, 2021.

W. Hwang, J. Yim, S. Park, and M. Seo. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv*, 2019.

R. Jia and P. Liang. Data recombination for neural semantic parsing. In *ACL*, 2016.

P. Kapanipathi, I. Abdelaziz, S. Ravishankar, S. Roukos, A. Gray, R. Fernandez Astudillo, M. Chang, C. Cornelio, S. Dana, A. Fokoue, D. Garg, A. Gliozzo, S. Gurajada, H. Karanam, N. Khan, D. Khandelwal, Y. Lee, Y. Li, F. Luus, N/ Makondo, N. Mihindukulasooriya, T. Naseem, S. Neelam, L. Popa, R. Gangi Reddy, R. Riegel, G. Rossiello, U. Sharma, S. Bhargav, and M. Yu. Leveraging abstract meaning representation for knowledge base question answering. In *Findings of ACL-IJCNLP*, 2021.

J. Krishnamurthy, P. Dasigi, and M. Gardner. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*, 2017.

Y. Lan and J. Jiang. Query graph generation for answering multi-hop complex questions from knowledge bases. In *ACL*, 2020.

Y. Lan, S. Wang, and J. Jiang. Knowledge base question answering with topic units. In *IJCAI*, 2019a.

Y. Lan, S. Wang, and J. Jiang. Multi-hop knowledge base question answering with an iterative sequence matching model. In *ICDM*, 2019b.

Y. Lan, G. He, J. Jiang, J. Jiang, W. Zhao, and J. Wen. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *IJCAI*, 2021a.

Y. Lan, G. He, J. Jiang, J. Jiang, X. Zhao, and J. Wen. Complex knowledge base question answering: A survey. *arXiv*, 2021b.

C. Lee, O. Polozov, and M. Richardson. Kaggledbqa: Realistic evaluation of text-to-sql parsers. In *ACL-IJCNLP*, 2021.

B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, 2020.

C. Liang, J. Berant, Q. Le, K. Forbus, and N. Lao. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *ACL*, 2017.

Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J. Lou. Tapex: Table pre-training via learning a neural sql executor. In *ICLR*, 2021.

L. Mo, A. Lewis, H. Sun, and M. White. Towards transparent interactive semantic parsing via step-by-step correction. In *Findings of ACL*, 2022.

N. Ngomo. 9th challenge on question answering over linked data (qald-9). *language*, 2018.

A. Perevalov, X. Yan, L. Kovriguina, L. Jiang, A. Both, and R. Usbeck. Knowledge graph question answering leaderboard: A community resource to prevent a replication crisis. *arXiv*, 2022.

K. Qin, Y. Wang, C. Li, K. Gunaratna, H. Jin, V. Pavlu, and J. Aslam. A complex kbqa system using multiple reasoning paths. *arXiv*, 2020.

Y. Qiu, K. Zhang, Y. Wang, L. Jin, X.and Bai, S. Guan, and X. Cheng. Hierarchical query graph generation for complex question answering over knowledge graph. In *CIKM*, 2020.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

S. Ravishankar, J. Thai, I. Abdelaziz, N. Mihidukulasooriya, T. Naseem, P. Kapanipathi, G. Rossilleo, and A. Fokoue. A two-stage approach towards generalization in knowledge base question answering. *arXiv*, 2021.

H. Ren, H. Dai, B. Dai, X. Chen, M. Yasunaga, H. Sun, D. Schuurmans, J. Leskovec, and D. Zhou. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *ICML*, 2021.

O. Rubin and J. Berant. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *NAACL*, June 2021.

A. Saha, V. Pahuja, M. Khapra, K. Sankaranarayanan, and S. Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI*, 2018.

T. Scholak, N. Schucher, and D. Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv*, 2021.

N. Schucher, S. Reddy, and H. de Vries. The power of prompt tuning for low-resource semantic parsing. In *ACL*, 2022.

J. Shi, S. Cao, L. Pan, L. Xiang, Y.and Hou, J. Li, H. Zhang, and B. He. Kqa pro: A large diagnostic dataset for complex question answering over knowledge base. *arXiv*, 2020.

T. Soru, E. Marx, A. Valdestilhas, D. Moussallem, G. Publio, and M. Saleem. Where is linked data in question answering over linked data? *arXiv*, 2020.

Y. Su, H. Sun, B. Sadler, M. Srivatsa, I. Gür, Z. Yan, and X. Yan. On generating characteristic-rich question sets for qa evaluation. In *EMNLP*, 2016.

Y. Su, A. Awadallah, M. Khabsa, P. Pantel, M. Gamon, and M. Encarnacion. Building natural language interfaces to web apis. In *CIKM*, 2017.

Y. Sun, L. Zhang, G. Cheng, and Y. Qu. Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases. In *AAAI*, 2020.

I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

K. Tai, R. Socher, and C. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCNLP*, 2015.

A. Talmor and J. Berant. The web as a knowledge-base for answering complex questions. In *NAACL*, 2018.

P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In *ISWC*, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, $\mathcal{L}$. Kaiser, and I. Polosukhin. Attention is all you need. *NIPS*, 2017.

D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 2014.

B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *ACL*, 2020.

T. Xie, C. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C. Wu, M. Zhong, P. Yin, S. Wang, et al. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv*, 2022.

J. Yang, H. Jiang, Q. Yin, D. Zhang, B. Yin, and D. Yang. Seqzero: Few-shot compositional semantic parsing with sequential prompts and zero-shot models. *arXiv*, 2022.

Z. Yao, Y. Su, H. Sun, and W. Yih. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. In *EMNLP-IJCNLP*, 2019.

X. Ye, S. Yavuz, K. Hashimoto, Y. Zhou, and C. Xiong. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv*, 2021.

W. Yih, M. Chang, X. He, and J. Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP*, 2015.

W. Yih, M. Richardson, C. Meek, M. Chang, and J. Suh. The value of semantic parse labeling for knowledge base question answering. In *ACL*, 2016.

X. Yin, D. Gromann, and S. Rudolph. Neural machine translating from natural language to sparql. *Future Generation Computer Systems*, 2021.

T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *EMNLP*, 2018.

T. Yu, C. Wu, X. Lin, Y. Tan, X. Yang, D. Radev, C. Xiong, et al. Grappa: Grammar-augmented pre-training for table semantic parsing. In *ICLR*, 2020.

H. Zafar, G. Napolitano, and J. Lehmann. Formal query generation for question answering over knowledge bases. In *ESWC*, 2018.

L. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005.

H. Zhang, J. Cai, J. Xu, and J. Wang. Complex question decomposition for semantic parsing. In *ACL*, 2019.