ArcaneQA: Dynamic Program Induction and Contextualized Encoding for Knowledge Base Question Answering

Yu Gu
The Ohio State University
Columbus, Ohio, USA
gu.826@osu.edu

Yu Su
The Ohio State University
Columbus, Ohio, USA
su.809@osu.edu

Abstract

Question answering on knowledge bases (KBQA) poses a unique challenge for semantic parsing research due to two intertwined challenges: *large search space* and *ambiguities in schema linking*. Conventional ranking-based KBQA models, which rely on a candidate enumeration step to reduce the search space, struggle with flexibility in predicting complicated queries and have impractical running time. In this paper, we present ArcaneQA, a novel generation-based model that addresses both the large search space and the schema linking challenges in a unified framework with two mutually boosting ingredients: *dynamic program induction* for tackling the large search space and *dynamic contextualized encoding* for schema linking. Experimental results on multiple popular KBQA datasets demonstrate the highly competitive performance of ArcaneQA in both effectiveness and efficiency.[1]

1 Introduction

Modern knowledge bases (KBs) contain a wealth of structured knowledge. For example, FREEBASE (Bollacker et al., 2008) contains over 45 million entities and 3 billion facts across more than 100 domains, while GOOGLE KNOWLEDGE GRAPH has amassed over 500 billion facts about 5 billion entities (Sullivan, 2020). Question answering on knowledge bases (KBQA) has emerged as a user-friendly solution to access the massive structured knowledge in KBs.

KBQA is commonly modeled as a semantic parsing problem (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005) with the goal of mapping a natural language question into a logical form that can be executed against the KB (Berant et al., 2013; Cai and Yates, 2013; Yih et al., 2015). Compared with other semantic parsing settings such as text-to-SQL parsing (Zhong et al., 2017; Yu et al., 2018), where the underlying data is moderate-sized, the massive scale and the broad-coverage schema of KBs makes KBQA a uniquely challenging setting for semantic parsing research.

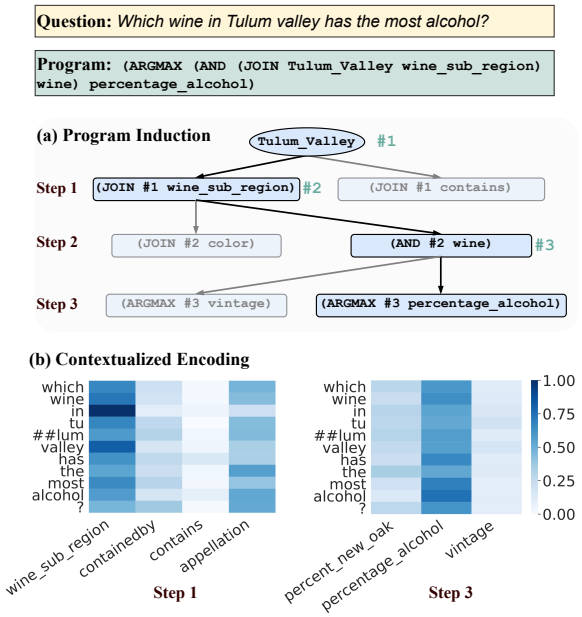

Figure 1: KBQA is commonly modeled as semantic parsing with the goal of mapping a question into an executable program. **(a)** A high-level illustration of our program induction procedure. The target program is induced by incrementally synthesizing a sequence of subprograms (`#1-3`). The execution of each subprogram can significantly reduce the search space of subsequent subprograms. **(b)** Alignments between question words and schema items at different steps achieved by a BERT encoder. A pre-trained language model like BERT can jointly encode the question and schema items to get the contextualized representation at each step, which further guides the search process.

The unique difficulty stems from two intertwined challenges: *large search space* and *ambiguities in schema linking*. On the one hand, transductive semantic parsing models that are highly effective in other semantic parsing settings (Dong and Lapata,

[1] Data and code: dki-lab/ArcaneQA

Proceedings of the 29th International Conference on Computational Linguistics, pages 1718–1731
October 12–17, 2022.

2016; Wang et al., 2020) struggle with the large vocabulary size and often generate logical forms (i.e., formal queries)[2] that are not faithful to the underlying KB (Gu et al., 2021; Xie et al., 2022). Therefore, a candidate enumeration and ranking approach is commonly adopted for KBQA (Berant and Liang, 2014; Yih et al., 2015; Abujabal et al., 2017; Lan et al., 2019a; Sun et al., 2020; Gu et al., 2021; Ye et al., 2021). However, these methods have to make various compromises on the complexity of admissible logical forms to deal with the large search space. Not only does this limit the type of answerable questions, but it also leads to impractical runtime performance due to the time-consuming candidate enumeration (Gu et al., 2021). On the other hand, schema linking,[3] i.e., mapping natural language to the corresponding schema items in the KB (e.g., in Figure 1, `wine_sub_region` is a linked schema items), is also a core challenge of KBQA. Compared with text-to-SQL parsing (Hwang et al., 2019; Zhang et al., 2019; Wang et al., 2020), the broad schema of KBs and the resulting ambiguity between schema items makes accurate schema linking more important and challenging for KBQA. Recent studies show that contextualized joint encoding of natural language questions and schema items with BERT (Devlin et al., 2019) can significantly boost the schema linking accuracy (Gu et al., 2021; Chen et al., 2021). However, existing methods still struggle with the large search space and need to encode a large number of schema items, which is detrimental to both accuracy and efficiency.

We present ArcaneQA (Dynamic Program Induction and Contextualized Encoding for Question Answering), a *generation-based* KBQA model that addresses both the large search space and the schema linking challenges in a unified framework. Compared with the predominant ranking-based KBQA models, our generation-based model can prune the search space on the fly and thus is more flexible to generate diverse queries without compromising the expressivity or complexity of answerable questions. Inspired by prior work (Dong and Lapata, 2016; Liang et al., 2017; Semantic Machines et al., 2020; Chen et al., 2021), we model KBQA using the encoder-decoder framework. However, instead of top-down decoding with grammar-level constraints as in prior work, which does not guarantee the faithfulness of the generated queries to the underlying KB, ArcaneQA performs *dynamic program induction* (Liang et al., 2017; Semantic Machines et al., 2020), where we incrementally synthesize a program by dynamically predicting a sequence of subprograms to answer a question; i.e., *bottom-up parsing* (Cheng et al., 2019; Rubin and Berant, 2021). Each subprogram is grounded to the KB and its grounding (i.e., denotation or execution results) can further guide an efficient search for faithful programs (see Figure 1(a)). In addition, we unify the meaning representation (MR) for programs in KBQA using S-expressions and support more diverse operations over the KB (e.g., numerical operations such as `COUNT/ARGMIN/ARGMAX` and diverse graph traversal operations).

At the same time, we employ pre-trained language models (PLMs) like BERT to jointly encode the question and schema items and get the contextualized representation of both, which implicitly links words to the corresponding schema items via self-attention. One unique feature to note is that *the encoding is also dynamic*: at each prediction step, only the set of admissible schema items determined by the dynamic program induction process needs to be encoded, which allows extracting the most relevant information from the question for each prediction step while avoiding the need to encode a large number of schema items. Figure 1(b) illustrates the contextualization of different steps via the attention heatmaps of BERT. In this example, the attention of each question word over candidate schema items serves as a strong indicator of the gold items for both steps (i.e., `wine_sub_region` for step 1 and `percentage_alcohol` for step 3). The two key ingredients of our model are *mutually boosting*: dynamic program induction significantly reduces the number of schema items that need to be encoded, while dynamic contextualized encoding intelligently guides the search process.

Our main contribution is as follows: a) We propose a novel generation-based KBQA model that is flexible to generate diverse complex queries while also being more efficient than ranking-based models. b) We propose a novel strategy to effectively employ PLMs to provide contextualized encoding for KBQA. c) We unify the meaning representation (MR) of different KBQA datasets and support

[2]We use the terms logical form, query, and program interchangeably across the paper.

[3]Semantic parsing implicitly entails two sub-tasks: *schema linking* and *composition*. There is not necessarily a dedicated step or component for schema linking. More commonly, the two sub-tasks are handled simultaneously.

more diverse operations. d) With our unified MR, we evaluate our model on three popular KBQA datasets and show highly competitive results.

2 Related Work

Ranking-Based KBQA. To handle the large search space in KBQA, existing studies typically rely on hand-crafted templates with a pre-specified maximum number of relations to enumerate candidate logical forms (Yih et al., 2015; Abujabal et al., 2017; Lan et al., 2019a; Bhutani et al., 2019, 2020), which suffers from limited expressivity and scalability. For example, Yih et al. (2015) limit the candidate programs to be a core relational chain, whose length is at most two, plus constraints. Ye et al. (2021) additionally adopts a post-generation module to revise the enumerated logical forms into more complicated ones, however, their method still heavily depends on the candidate enumeration step. In addition, the time-consuming candidate enumeration results in impractical online inference time for ranking-based models. In contrast, ArcaneQA obviates the need for candidate enumeration by pruning the search space on the fly and thus can generate more diverse and complicated programs within practical running time.

Generation-Based KBQA. To relax the restriction on candidate enumeration, some recent efforts are made to reduce the search space using beam search (Lan et al., 2019b; Chen et al., 2019; Lan and Jiang, 2020), however, Lan et al. (2019b) and Chen et al. (2019) can only generate programs of path structure, while Lan and Jiang (2020) follow the query graph structure proposed by Yih et al. (2015). A few recent studies (Liang et al., 2017; Chen et al., 2021) formulate semantic parsing over the KB as sequence transduction using encoder-decoder models to enable more flexible generation. Chen et al. (2021) apply schema-level constraints to eliminate ill-formed programs from the search space, however, they do not guarantee the faithfulness of predicted programs. Similar to Liang et al. (2017), our dynamic program induction uses KB contents-level constraints to ensure the faithfulness of generated programs, but we extend it to handle more complex and diverse questions and also use it jointly with dynamic contextualized encoding.

Using PLMs in Semantic Parsing. PLMs have been widely applied in many semantic parsing tasks, typically being used to jointly encode the input question and schema items(Hwang et al., 2019; Zhang et al., 2019; Wang et al., 2020; Scholak et al., 2021). However, PLMs have been under-exploited in KBQA. One major difficulty of using PLMs in KBQA lies in the high volume of schema items in a KB; simply concatenating all schema items with the input question for joint encoding, as commonly done in text-to-SQL parsing, will vastly exceed PLMs' maximum input length. Existing KBQA models either use PLMs to provide features for downstream classifiers(Lan and Jiang, 2020; Sun et al., 2020) or adopts a pipeline design to identify a smaller set of schema items beforehand and only use PLMs to encode these identified items (Gu et al., 2021; Chen et al., 2021), which can lead to error propagation. By comparison, ArcaneQA can fully exploit PLMs to provide contextualized representation for the question and schema items dynamically, where only the most relevant schema items are encoded at each step. More recently, Ye et al. (2021) use T5 (Raffel et al., 2019) to output a new program given a program as input, while T5's decoder generates free-formed text and does not always produce faithful programs. By contrast, ArcaneQA only uses PLMs for encoding and uses its customized decoder with a faithfulness guarantee.

3 Background

Knowledge Base. A knowledge base $\mathcal{K}$ consists of a set of relational triplets $\mathcal{K}_r \subset \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L})$ and a set of class assertions $\mathcal{K}_c \subset \mathcal{E} \times \mathcal{C}$, where $\mathcal{C}$ is a set of classes, $\mathcal{E}$ is a set of entities, $\mathcal{L}$ is a set of literals and $\mathcal{R}$ is a set of binary relations. Elements in $\mathcal{C}$ and $\mathcal{R}$ are also called the schema items of $\mathcal{K}$.

Meaning Representation for KBQA. Prior work adopt different meaning representations to represent logical forms for KBQA. For example, Yih et al. (2015) use graph query, which represents a program as a core relation chain with (optionally) some entity constraints. Cai and Yates (2013) use λ-Calculus as their meaning representation. In this paper, we follow Gu et al. (2021) to use S-expressions as our meaning representation due to their expressivity and simplicity. To support more diverse operations over the KB, we extend their definitions with two additional functions `CONS` and `TC`, which are used to support constraints with implicit entities and temporal constraints respectively (see details in Appendix A). For implicit entities, consider the question *"What was Elie Wiesel's father's name?"*, whose target query involves two entities: `Elie_Wiesel` and

Male. The entity Male is an implicit constraint rather than a named entity,[4] and it is used as an argument of CONS in the target logical form: (CONS (JOIN people.person.children Elie_Wiesel) people.person.gender Male). TC works in a similar way, with the difference being that the constraint should be a temporal expression (e.g., 2015-08-10) rather than an implicit entity.

4 Approach

4.1 Overview

The core idea of our generation-based model is to gradually expand a subprogram (i.e., a partial query) into the finalized target program, instead of enumerating all possible finalized programs from the KB directly, which suffers from combinatorial explosion. There are two common strategies to instantiate the idea of gradual subprogram expansion, depending on the type of meaning representation being used. For a graph-like meaning representation, we can directly perform graph search over the KB to expand a subprogram (Chen et al., 2019; Lan and Jiang, 2020). Also, we can linearize a program into a sequence of tokens and perform decoding in the token space (Liang et al., 2017; Scholak et al., 2021). Because S-expressions can be easily converted into sequences of tokens, we choose to follow the second strategy and take advantage of the encoder-decoder framework, which has been a de facto choice for many semantic parsing tasks. Concretely, ArcaneQA learns to synthesize the target program by dynamically generating a sequence of subprograms token by token until predicting $\langle EOS \rangle$, where each subsequent subprogram is an expansion from one or more preceding subprograms (denoted as parameters in the subsequent subprogram). Formally, the goal is to map an input question $q = x_1, ..., x_{|q|}$ to a sequence of subprograms $o = o_1^1, ..., o_{|o^1|}^1, ..., o_1^k, ..., o_{|o^k|}^k = y_1, ..., y_{|o|}$, where k is the number of total subprograms and $|o| = \sum_{i=1}^{k} |o^i|$. We base ArcaneQA on Seq2Seq with attention (Sutskever et al., 2014; Bahdanau et al., 2015), in which the conditional probability $p(o|q)$ is decomposed as:

$$p(o|q) = \prod_{t=1}^{|o|} p(y_t|y_{<t}, q), \tag{1}$$

where each token y_t is either a token from the vocabulary $\mathcal{V}$ or an intermediate subprogram from the set $\mathcal{S}$ storing all previously generated subprograms. $\mathcal{V}$ comprises all schema items in $\mathcal{K}$, syntactic symbols in S-expressions (i.e., parentheses and function names), and the special token $\langle EOS \rangle$. $\mathcal{S}$ initially contains the identified entities in the question (e.g., #1 in Figure 1). Every time a subprogram is predicted, it is executed and added to $\mathcal{S}$ (e.g., #2 in Figure 1).

ArcaneQA builds on two key ideas: *dynamic program induction* and *dynamic contextualized encoding* (see Figure 2). At each decoding step, ArcaneQA only makes a prediction from a small set of admissible tokens instead of the entire vocabulary. This is achieved by the dynamic program induction framework (subsection 4.2), which effectively prunes the search space by orders of magnitude and guarantees that the predicted programs are faithful to the KB. In addition, we dynamically apply BERT to provide contextualized joint encoding for both the question and admissible tokens at each decoding step (subsection 4.3). In this way, we allow the contextualized encoding to only focus on the most relevant information without introducing noise from irrelevant tokens.

4.2 Dynamic Program Induction

Dynamic program induction capitalizes on the idea that a complicated program can be gradually expanded from a list of subprograms. To ensure the expanded program is faithful to the KB, we query the KB with a subprogram to expand and a function defined in Table 4 to get a set of admissible actions (tokens). For example, in Figure 2, given #1 and the function JOIN, the admissible actions are defined by predicting a relation connecting to the execution result of #1 (i.e., Tulum_Valley), and there are only four relations to choose from (e.g., appellation and wine_sub_region). Table 1 shows a comprehensive description of expansion rules for different functions. With these rules, ArcaneQA can greatly reduce the search space for semantic parsing over the KB dynamically. The reduced candidate space further allows us to perform dynamic contextualized encoding (subsection 4.3).

[4] WEBQSP is the only dataset we consider that has this feature. Though there might be a more systematic way to differentiate implicit entities from named entities, we choose an expedient way to collect implicit entities from the training data according to whether an entity is explicitly mentioned in the question.

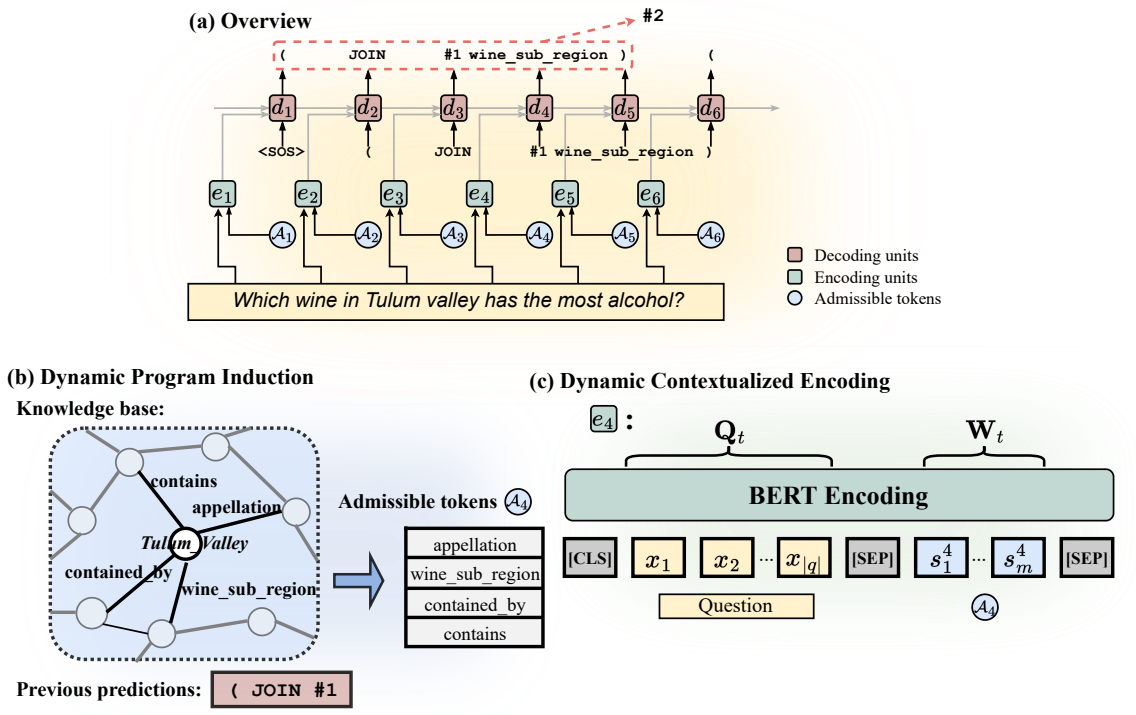

Figure 2: **(a)** Overview of ArcaneQA. ArcaneQA synthesizes the target program by iteratively predicting a sequence of subprograms. **(b)** At each step, it makes a prediction from a small set of admissible tokens $\mathcal{A}$ dynamically determined based on the execution of previous subprograms (for faithfulness to the KB) as well as the grammar (for well-formedness). **(c)** ArcaneQA also leverages BERT to provide dynamic contextualized encoding of the question and the admissible tokens at each step, which enables implicit schema linking and guides the dynamic program induction process.

Within our encoder-decoder framework, this idea is implemented using constrained decoding (Liang et al., 2017; Scholak et al., 2021), i.e., at each decoding step, a small set of admissible tokens from the vocabulary is determined based on the decoding history following predefined rules. The expansion rules in Table 1 have already comprised part of our rules for constrained decoding. In addition, several straightforward grammar rules are applied to ensure that the generated programs are well-formed. For instance, after predicting "(", the admissible tokens for the next step can only be a function name. After predicting a function name, the decoder can only choose a preceding subprogram to expand. After predicting ")", the admissible tokens for next step can only be either "(", indicating the start of a new subprogram, or "$\langle EOS \rangle$", denoting termination. The decoding process can be viewed as a *sequential decision-making process*, which decomposes the task of finding a program from the enormous search space into making decisions from a sequence of smaller search spaces.

4.3 Dynamic Contextualized Encoding

In semantic parsing, PLMs have typically been used to jointly encode the input question and all schema items via concatenation (Hwang et al., 2019; Zhang et al., 2019; Wang et al., 2020). However, direct concatenation is not feasible for KBQA due to a large number of schema items. Instead of obtaining a static representation for the question and items from $\mathcal{V}$ before decoding (Gu et al., 2021; Chen et al., 2021), we propose to do dynamic contextualized encoding at each decoding step; for each step, we use BERT to jointly encode the question and only the admissible tokens from $\mathcal{V}$. ArcaneQA's dynamic program induction vastly reduces the number of candidate tokens at each step and allows us to concatenate the question and the admissible tokens into a compact sequence:[5]

$$\text{[CLS]}, x_1, ..., x_{|q|}, \text{[SEP]}, s_1^t, ..., s_m^t, \text{[SEP]}$$

where $\{s_i^t\} \subset \mathcal{V}$ are admissible tokens at

[5]We omit the wordpieces tokenization here for brevity.

Current function	Admissible actions
JOIN	$\{r \vert h \in \#, (h, r, t) \in \mathcal{K}_r\}$
AND	$\{v \vert v \in \mathcal{S}, v \cap \# \neq \emptyset\} \cup \{c \vert e \in \#, (e, c) \in \mathcal{K}_c\}$
ARGMAX/ARGMIN	$\{r \vert h \in \#, t \in \mathcal{L}, (h, r, t) \in \mathcal{K}_r\}$
LT(LE/GT/GE)	$\{r \vert t < (\leq / > / \geq)\#, (h, r, t) \in \mathcal{K}_r\}$
COUNT	$\{)\}$
CONS	$\{(r, t) \vert h \in \#, (h, r, t) \in \mathcal{K}_r\}$
TC	$\{(r, t) \vert h \in \#, (h, r, t) \in \mathcal{K}_r, t \in \mathcal{L}$ is a temporal expression$\}$

Table 1: A set of rules to expand a preceding subprogram given a function. The execution of the subprogram is denoted as #. These expansion rules reduce the search space significantly with a faithfulness guarantee. COUNT takes no other argument, so the only admissible token is ")".

step t and $|\{s_i^t\}| = m$. After feeding the concatenated sequence to BERT, we obtain the question representation $\mathbf{Q_t} = (\mathbf{x_1}, ..., \mathbf{x_q})$ by further feeding the outputs from BERT to an LSTM encoder. For each admissible token, we represent it by averaging BERT outputs corresponding to its wordpieces. In this way, we also obtain the embedding matrix $\mathbf{W_t} \in \mathbb{R}^{m \times d}$, where each row corresponds to the embedding of an admissible token. The contextualized representation $\mathbf{Q_t}$ and $\mathbf{W_t}$ are both dynamically computed at each time step. Words and corresponding schema items are implicitly linked to each other via BERT's self-attention.

4.4 Decoding

We use an LSTM decoder. At decoding step t, given the hidden state $\mathbf{h_{t-1}}$ and input $\mathbf{c_{t-1}}$, we obtain the updated hidden state $\mathbf{h_t}$ by:

$$\mathbf{h_t} = \text{LSTM}_\theta(\mathbf{h_{t-1}}, \mathbf{c_{t-1}}) \quad (2)$$

where our LSTM decoder is parameterized by θ.

With $\mathbf{h_t}$ and $\mathbf{W_t}$—the embedding matrix of admissible tokens (determined by dynamic program induction)—we obtain the probability of generating a token from the admissible tokens:

$$p(y_t = s_{ti} | q, y_{<t}) = [\text{Softmax}(\mathbf{W_t h_t})]_i \quad (3)$$

The input $\mathbf{c_t}$ for the next step is obtained via the concatenation of the contextualized embedding of the current output token and the weighted representation of the question based on attention:

$$\mathbf{a_t} = \text{softmax}(\mathbf{Q_t h_t}) \quad (4)$$

$$\mathbf{q_t} = (\mathbf{a_t})^T \mathbf{Q_t} \quad (5)$$

$$\mathbf{c_t} = [[\mathbf{W_t}]_j; \mathbf{q_t}] \quad (6)$$

where ; denotes concatenation, and j denotes the index of the predicted y_t in $\mathbf{W_t}$.

4.5 Training and Inference

We train ArcaneQA with question-program pairs using cross entropy loss. The model learns to maximize the probability of predicting the gold token out of a small set of admissible tokens at each step, which is different from training a conventional Seq2Seq model using a static vocabulary.

During inference, ArcaneQA assumes an entity linker to identify a set of entities from the question at the beginning of program induction. However, the entity linker may identify false entities. To deal with it, ArcaneQA initiate its decoding process with different hypotheses from the set of entities. Basically, it tries out all possible combinations of the identified entities (i.e., the power set of the identified entities), considering that our entity linker normally can only identify no more than two entities from a question.

5 Experimental Setup

Datasets. We evaluate ArcaneQA on three KBQA datasets covering diverse KB queries.

GRAILQA (Gu et al., 2021) is a large-scale KBQA dataset that contains complex questions with various functions, including comparatives, superlatives, and counting. It evaluates the generalizability of KBQA at three levels: i.i.d., compositional and zero-shot.

GRAPHQ (Su et al., 2016) also contains questions of diverse nature. It is particularly challenging because it exclusively focuses on non-i.i.d. generalization.[6]

WEBQSP(Yih et al., 2016) is a clean subset of WEBQ (Berant et al., 2013) with annotated logical forms.

[6] GRAPHQ originally uses FREEBASE (version 2013-07) as their KB, while GRAILQA and WEBQ use FREEBASE (version 2015-08-09). In Gu et al. (2021), programs in GRAPHQ are converted into the corresponding FREEBASE 2015-08-09 version, and we will use this version in our experiments.

All questions in it are collected from Google query logs, featuring more realistic and complicated information needs such as questions with temporal constraints.

The total number of questions in GRAILQA, GRAPHQ, and WEBQ is 64,331, 5,166, and 4,737 respectively.

Evaluation Metrics. For GRAILQA, we use their official evaluation script with two metrics, EM, i.e., program exact match accuracy, and F1, which is computed based on the predicted and the gold answer set. For GRAPHQ and WEBQSP, we follow the standard practice and report F1.

Models for Comparison. We compare ArcaneQA with the previous best-performing models on three different datasets. For GRAILQA and WEBQSP, the state-of-the-art model is **RnG-KBQA** (Ye et al., 2021). Though RnG-KBQA uses T5 to decode the target program as unconstrained sequence transduction, it still heavily depends on candidate enumeration as a prerequisite. Therefore, it is not a generation-based model like ours. **ReTraCk** (Chen et al., 2021) is the state-of-the-art generation-based model on GRAILQA which poses grammar-level constraints to the decoder to generate well-formed but unnecessarily faithful programs. For GRAPHQ, the ranking-based model **SPARQA** (Sun et al., 2020) has achieved the best results so far. It uses BERT as a feature extractor for downstream classifiers. In addition to the state-of-the-art models, we also compare ArcaneQA with **BERT+Transduction** and **BERT+Ranking** (Gu et al., 2021), which are two baseline models on GRAILQA that enhance a vanilla Seq2Seq model with BERT to perform generation and ranking respectively.

Implementation. Our models are implemented using PyTorch and AllenNLP (Gardner et al., 2018). For BERT, we use the bert-base-uncased version provided by HuggingFace. For more details about implementation and hyper-parameters, we refer the reader to Appendix B.

6 Results

6.1 Overall Evaluation

We show the overall results in Table 2 (for dev set results see Appendix C). ArcaneQA achieves the state-of-the-art performance on both GRAPHQ and WEBQSP. For GRAPHQ, there are 188 questions in GRAPHQ's test set that cannot be converted into FREEBASE 2015-08-09 version, so we treat the F1 of all those questions as 0 following Gu et al. (2021), while the numbers in the parentheses are the actual F1 on the test set if we exclude those questions. ArcaneQA significantly outperforms the prior art by over 10%. The improvement over SPARQA shows the advantage of using PLMs for contextualized joint encoding instead of just providing features for ranking. On both WEBQSP and GRAILQA, ArcaneQA also achieves the best performance or performs on par with the prior art in terms of F1. It outperforms ReTraCk by 4.3% and 1.9% (using the same entity linking results) on WEBQSP and GRAILQA respectively, suggesting that ArcaneQA can more effectively reduce the search space with dynamic program induction compared with ReTraCk's grammar-based decoding. Also, our model performs on par with the previous state-of-the-art RnG-KBQA (i.e., same numbers on WEBQSP, while 0.7% lower on GRAILQA). However, ArcaneQA under-performs RnG-KBQA in EM on GRAILQA. The overall EM of ArcaneQA is lower than RnG-KBQA by 5%, and the gap on zero-shot generalization is even larger (i.e., around 10%), despite the comparable numbers in F1. This can be explained by that ArcaneQA learns to predict a program in a more flexible way and may potentially find some novel structures. This may further be supported by the observation that ArcaneQA performs better than RnG-KBQA on compositional generalization, which requires KBQA models to generalize to unseen query structures during training. Overall, the results demonstrate ArcaneQA's flexibility in handling KBQA scenarios of different natures.

6.2 In-Depth Analyses

To gain more insights into ArcaneQA's strong performance, we conduct in-depth analyses on the two key designs of ArcaneQA.

Dynamic Program Induction. One vanilla implementation of ArcaneQA without dynamic program induction is BERT+Transduction, i.e., its search space and vocabulary during decoding is independent of previous predictions. As shown in Table 2a, when using the same entity linking results, ArcaneQA outperforms BERT+Transduction by 30.4% in overall F1 and is twice as good on zero-shot generalization. One major weakness of

Model	Overall		I.I.D.		Compositional		Zero-shot	
	EM	F1	EM	F1	EM	F1	EM	F1
QGG* (Lan and Jiang, 2020)	–	36.7	–	40.5	–	33.0	–	36.6
BERT+Transduction* (Gu et al., 2021)	33.3	36.8	51.8	53.9	31.0	36.0	25.7	29.3
BERT+Ranking* (Gu et al., 2021)	50.6	58.0	59.9	67.0	45.5	53.9	48.6	55.7
ReTraCk (Chen et al., 2021)	58.1	65.3	84.4	87.5	61.5	70.9	44.6	52.5
RnG-KBQA* (Ye et al., 2021)	61.4	67.4	78.0	81.8	55.0	63.2	56.7	63.0
ArcaneQA*	58.8	67.2	77.8	81.6	58.0	66.1	50.4	61.8
RnG-KBQA (Ye et al., 2021)	**68.8**	**74.4**	**86.2**	**89.0**	63.8	71.2	**63.0**	**69.2**
ArcaneQA	63.8	73.7	85.6	88.9	**65.8**	**75.3**	52.9	66.0
w/o contextualized encoding	49.7	59.1	77.6	82.1	50.5	59.4	36.5	48.5

(a) GRAILQA

Model	F1
UDEPLAMBDA (Reddy et al., 2017)	17.7
PARA4QA (Dong et al., 2017)	20.4
SPARQA (Sun et al., 2020)	21.5
BERT+Ranking (Gu et al., 2021)	25.0 (27.0)
ArcaneQA	**31.8** (**34.3**)
w/o contextualized encoding	20.7 (22.4)

(b) GRAPHQ

Model	F1
NSM (Liang et al., 2017)	69.0
KBQA-GST (Lan et al., 2019a)	67.9
TextRay (Bhutani et al., 2019)	60.3
QGG (Lan and Jiang, 2020)	74.0
ReTraCk (Chen et al., 2021)	71.0
CBR (Das et al., 2021)	72.8
RnG-KBQA (Ye et al., 2021)	**75.6** (74.5♯)
ArcaneQA	**75.6** (**75.6**♯)
w/o contextualized encoding	68.8

(c) WEBQSP

Table 2: Overall results on three datasets. ArcaneQA follows entity linking results from previous methods (i.e., RnG-KBQA's results on GRAILQA, QGG's results on WEBQSP, and Gu et al. (2021)'s results on GRAPHQ) for fair comparison. Model names with * indicate using the baseline entity linking results on GRAILQA. ♯ In addition to using WEBQSP's official evaluation script, which sometimes considers multiple target parses for a question, we also report the performance when only the top-1 target parses are considered.

BERT+Transduction is that it predicts many programs that are not faithful to the KB, executing which will lead to empty answers. Note that post-hoc filtering by execution (Wang et al., 2018) can only help to a limited degree due to the KB's broad schema, while this type of mistake is rooted out in ArcaneQA by design.

Different from our search space pruning achieved with dynamic program induction, ranking-based models such as BERT+Ranking prunes unfaithful programs from their search space by ranking a set of faithful programs enumerated from the KB. These models typically make compromises on the complexity and diversity of programs during candidate enumeration. We break down the performance of ArcaneQA on GRAILQA's validation set in terms of question complexity and function types and show the fine-grained results in Table 3. The comparison with BERT-Ranking demonstrates the scalability and flexibility of our dynamic program induction. We also compare with RnG-KBQA, which adopts exactly the same candidate enumeration module as BERT+Ranking, but it is enhanced with a T5-based revision module to edit the enumerated programs into more diverse ones. We observe that RnG-KBQA performs uniformly well across different programs except for programs with superlative functions (i.e., ARGMAX/ARGMIN), i.e., the F1 of it is lower than ArcaneQA by over 50%. This is because in their candidate generation step, there is no superlative function enumerated. Despite the effectiveness of their T5-based revision, their performance still heavily depends on the diversity of candidate enumeration, which restricts the flexibility of their method.

Dynamic Contextualized Encoding. To show the key role of dynamic contextualized encoding, we use GloVe (Pennington et al., 2014) to provide non-contextualized embeddings for both questions and tokens in $\mathcal{V}$. We fix GloVe embeddings during training to make the model less biased to the training distribution (Gu et al., 2021) for GRAILQA and GRAPHQ, which address non-i.i.d. generalization, while for WEBQSP, we also update the word embeddings during training. Results in Table 2a show the importance of dynamic contextualized encoding, i.e., without contextualized encoding, the overall F1 decreases by 14.6%, 11.1%, and 6.5% on three datasets respectively. We also notice that dynamic contextualized encoding is more criti-

Function	None	Count	Comparative	Superlative
BERT+Ranking	59.1/66.0	43.0/53.2	0.0/14.5	0/6.0
RnG-KBQA	**77.5/81.8**	**73.0/77.5**	**55.1/76.0**	13.8/22.3
ArcaneQA	70.8/77.8	62.5/68.2	54.5/75.7	**70.5/75.6**
# of relations	**1**	**2**	**3**	**4**
BERT+Ranking	57.4/61.5	39.8/54.7	0.0/22.9	0.0/25.0
RnG-KBQA	**75.7**/79.2	**65.4/74.8**	**28.6/44.4**	**100.0/100.0**
ArcaneQA	74.9/**80.9**	59.9/71.1	27.6/37.7	**100.0/100.0**

Table 3: Fine-grained results (EM/F1) on GRAILQA's dev set. **None** denotes programs with only AND and JOIN.

cal for non-i.i.d. generalization, i.e., on GRAILQA the F1 on i.i.d. generalization only decreases by 6.8%, while it decreases by 15.9% and 17.5% on compositional and zero-shot generalization. Without contextualized encoding, identifying the correct schema items from the KB in non-i.i.d. setting is particularly challenging. Schema linking powered by dynamic contextualized encoding is the key to non-i.i.d. generalization, which is a long-term goal of KBQA.

6.3 Efficiency Analysis

We compare the running time of ArcaneQA and ranking-based models in the online mode (i.e., no offline caching) to mimic the real application scenario. To make the comparison fair, we configure all models to interact with the KB via the same Virtuoso SPARQL endpoint. We run each model on 1,000 randomly sampled questions and report the average running time per question on a GTX 2080 Ti card. As shown below, our model is faster than BERT+Ranking and RnG-KBQA by an order of magnitude, because ArcaneQA dynamically prunes the search space and does not run the time-consuming queries for enumerating two-hop candidates.

	BERT+Ranking	RnG-KBQA	ArcaneQA
Time (s)	115.5	82.1	5.6

7 Conclusions

We present a novel generation-based KBQA model, ArcaneQA, which simultaneously addresses the large search space and schema linking challenges in KBQA with dynamic program induction and dynamic contextualized encoding. Experimental results on several popular datasets demonstrate the advantages of ArcaneQA in both effectiveness and efficiency. In the future, we will focus on developing generation-based KBQA models with stronger zero-shot generalizability. In addition, exploring other pre-trained language models such as T5 (Raffel et al., 2019) for generation-based KBQA is also an interesting direction.

Acknowledgement

The authors would like to thank the colleagues from the OSU NLP group and the anonymous reviewers for their thoughtful comments. This research was supported by NSF OAC 2112606.

References

Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1191–1200, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425.

Nikita Bhutani, Xinyi Zheng, and HV Jagadish. 2019. Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 739–748.

Nikita Bhutani, Xinyi Zheng, Kun Qian, Yunyao Li, and H. Jagadish. 2020. Answering complex questions by combining information from curated and extracted knowledge bases. In *Proceedings of the First Workshop on Natural Language Interfaces*, pages 1–10, Online. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing Freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338, Atlanta, Georgia, USA. Association for Computational Linguistics.

Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 2021. Retrack: A flexible and efficient framework for knowledge base question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 325–336.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. Uhop: An unrestricted-hop relation extraction framework for knowledge-based question answering. *arXiv preprint arXiv:1904.01246*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. *arXiv preprint arXiv:2104.08762*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886, Copenhagen, Denmark. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.

Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019a. Knowledge base question answering with topic units.(2019). In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 5046–5052.

Yunshi Lan, Shuohang Wang, and Jing Jiang. 2019b. Multi-hop knowledge base question answering with an iterative sequence matching model. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 359–368. IEEE.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.

Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 311–324, Online. Association for Computational Linguistics.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*.

Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for QA evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas. Association for Computational Linguistics.

Danny Sullivan. 2020. A reintroduction to our Knowledge Graph and knowledge panels. blog.google.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of the Thirty-Fourth National Conference on Artificial Intelligence*, AAAI'20. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

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, Online. Association for Computational Linguistics.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv preprint arXiv:2109.08678*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris 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 (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational 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 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*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial intelligence*, pages 1050–1055.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev.

2019. Editing-based sql query generation for cross-domain context-dependent questions. *arXiv preprint arXiv:1909.00786*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Meaning Representation

We provide a detailed description of our defined functions for S-expressions in Table 4. We provide annotations in S-expressions for several KBQA datasets, including WEBQSP, GRAPHQ, and COMPLEXWEBQ (which we did not use for experiments). All data files annotated by us can be found in our Github Repo.

B Implementation Details

B.1 Entity Linking Results

For GRAILQA, we use the entity linking results provided by Ye et al. (2021); for GRAPHQ, we use the entity linking results provided by Gu et al. (2021); for WEBQSP, we follow the entity linking results provided by (Lan and Jiang, 2020). In addition, we find that answer types can be a strong clue for GRAILQA, so we also predict a set of FREEBASE classes for GRAILQA as a special type of entity using a BERT-based classifier. All entity linking results can be found in our Github Repo.

B.2 Entity Anonymization

After identifying a set of entities, we do entity anonymization for WEBQ, i.e., we replace the entity mention with the type of the corresponding entity. For example, mention "*Barack Obama*" will be replaced by "*US president*". However, the entity linker might identify some false positive mentions, and anonymizing these mentions would lead to some critical information loss. To address this problem, we identify a set of common false positive mentions that contain important information about the question in training data. Such words include "government", "zip", etc. For mentions include these words, we do not do anonymization. Doing entity anonymization is a common practice on WEBQ, which can normally bring some gain of around 1 to 2 percent in F1, while for GRAILQA and GRAPHQ, we did not observe any improvement, so we keep the original entity mentions for these two datasets.

B.3 Hyper Parameters

For ArcaneQA, we are only able to train our model with batch size 1 due to the memory consumption, so we choose a workaround to set the number of gradient accumulations to 16. We use Adam optimizer with an initial learning rate of 0.001 to update our own parameters in BERT-based models. For BERT's parameters, we fine-tune them with a learning rate of 2e-5. For ArcaneQA w/o BERT, we train it with batch size 32 and an initial learning rate of 0.001 using Adam optimizer. For both models, the hidden sizes of both encoder and decoder are set to 768, and the dropout rate is set to 0.5. All hyper-parameters are manually tuned according to the validation accuracy on the development set. specifically, we do manual hyper-parameter search from [1e-5, 2e-5, 3e-5], [8, 16, 32], [0.0, 0.2, 0.5] to tune the learning rate of fine-tuning BERT, steps of gradient accumulation and dropout rate respectively.

B.4 Number of Model Parameters

Total numbers of trainable parameters of ArcaneQA and ArcaneQA w/o BERT are 123,652,608 and 261,900 respectively. The reason that the trainable parameters of ArcaneQA w/o BERT are so few is that we freeze the GloVe embeddings for non-i.i.d. generalization. The number of parameters becomes 121,205,100 if we take the GloVe embeddings into account.

B.5 Other Details

We summarize some other details in our implementation that are critical to reproduction.

We identify the literals in GRAILQA and GRAPHQ using hand-crafted regular expressions. There are two types of literals, i.e., date time and numerical value. Our regular expressions can identify around 98% of all literals.

During dynamic program induction of ArcaneQA, we follow the rules in Table 1 to run SPARQL queries to retrieve the admissible schema items. However, in some rare cases, the execution of a subprogram may contain a tremendous number of entities For example, the execution of `(JOIN USA people.person.nationality)` contains over 500,000 entities, and running SPARQL queries for all entities in them is infeasible. As a result, we only run SPARQL queries for 100 entities sampled from the execution results. One better choice could be to use some more efficient indexing to query the KB instead of using SPARQL.

We construct the vocabulary $\mathcal{V}$ for different datasets in different ways. For GRAILQA, following Gu et al. (2021), we construct the vocabulary using schema items from FREEBASECOMMONS. For GRAPHQ, we construct the vocabulary using schema items from the entire FREEBASE. For WEBQ, because it evaluates i.i.d. generalization, so we construct the vocabulary from its training data.

Function	Arguments	Returns
JOIN	a set of entities $u \subset (\mathcal{E} \cup \mathcal{L})$ and a relation $r \in \mathcal{R}$	all entities connecting to any $e \in u$ via r
AND	two set of entities $u1 \subset \mathcal{E}$ and $u2 \subset \mathcal{E}$	the intersection of two entities sets.
ARGMAX/ARGMIN	a set of entities $u \subset \mathcal{E}$ and a numerical relation $r \in \mathcal{R}$	a set of entities from u with the maximum/minimum value for r
LT(LE/GT/GE)	a numerical value $u \subset \mathcal{L}$ and a numerical relation $r \in \mathcal{R}$	all entities with a value $< (\leq / > / \geq) u$ for relation r
COUNT	a set of entities $u \subset \mathcal{E}$	the number of entities in u
CONS	a set of entities a set of entities $u \subset \mathcal{E}$, a relation $r \in \mathcal{R}$, and a constraint $c \in (\mathcal{E} \cup \mathcal{L})$	all $e \in u$ satisfying $(e, r, c) \in \mathcal{K}_r$
TC	a set of entities a set of entities $u \subset \mathcal{E}$, a relation $r \in \mathcal{R}$, and a temporal constraint $c \in \mathcal{L}$	all $e \in u$ satisfying $(e, r, c) \in \mathcal{K}_r$

Table 4: Detailed descriptions of functions defined in our S-expressions. We extend the definitions in Gu et al. (2021) by introducing two new functions CONS and TC. Also, we remove the function R and instead represent the inverse of a relation by adding a suffix "_inv" to it. Note that, for arguments in AND function, a class $c \in \mathcal{C}$ can also indicate a set of entities which fall into c.

	Overall		I.I.D.		Compositional		Zero-shot	
Model	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
BERT+Ranking (Gu et al., 2021)	51.0	58.4	58.6	66.1	40.9	48.1	51.8	59.2
RnG-KBQA (Ye et al., 2021)	**71.4**	76.8	**86.7**	**89.0**	61.7	68.9	**68.8**	**74.7**
ArcaneQA	69.5	**76.9**	86.1	89.2	**65.5**	**73.9**	64.0	72.8

Table 5: The results on the validation set of GRAILQA. The overall trend is basically consistent with the test set.

C Results on the Validation Set of GRAILQA

We show the results of ArcaneQA, BERT+Ranking, and RnG-KBQA on the validation set of GRAILQA in Table 5. We observe that ArcaneQA achieves a better F1 than RnG-KBQA. Overall, the trend is consistent with the test set. We also observe that the EM of ArcaneQA on zero-shot generalization is significantly higher than the test set, which is interesting and remains for further investigation.