Bring Your Own KG: Self-Supervised Program Synthesis for Zero-Shot KGQA

Anonymous ACL submission

Abstract

We present BYOKG, a universal question-answering (QA) system that can operate on *any* knowledge graph (KG), requires no human-annotated training data, and can be ready to use within a day—attributes that are out-of-scope for current KGQA systems. BYOKG draws inspiration from the remarkable ability of humans to comprehend information present in an unseen KG through exploration—starting at random nodes, inspecting the labels of adjacent nodes and edges, and combining them with their prior world knowledge. Exploration in BYOKG leverages an LLM-backed symbolic agent that generates a diverse set of query-program exemplars, which are then used to ground a retrieval-augmented reasoning procedure to synthesize programs for arbitrary questions. BYOKG is effective over both small- and large-scale graphs, showing dramatic gains in zero-shot QA accuracy of 27.89 and 59.88 F1 on GrailQA and MetaQA, respectively. We further find that performance of BYOKG reliably improves with continued exploration as well as improvements in the base LLM, notably outperforming a state-of-the-art *fine-tuned* model by 7.08 F1 on a sub-sampled zero-shot split of GrailQA. Lastly, we verify our universality claim by evaluating BYOKG on a domain-specific materials science KG and show that it improves zero-shot performance by 46.33 F1.

1 Introduction

The ability to query structured data stores such as knowledge graphs (KGQA) via natural language is crucial for making the information within them accessible (Liang, 2016; Das, 2022). However, most prior works that aim to create such interfaces assume the availability of *some* training data (query-program pairs) (Talmor and Berant, 2018; Keysers et al., 2020; Gu et al., 2021; Dutt et al., 2023a; Sen et al., 2023), which, in practice, might be unrealistic. For example, in scientific domains such as materials science and clinical decision-making, training data may be completely unavailable due to high collection costs or stringent privacy regulations (Sima et al., 2022). Further, even when training data is available, models trained on one dataset may not generalize o.o.d. to other datasets of the *same* KG (Khosla et al., 2023).

In this work, we, therefore, set out to answer the following question—can we develop a universal QA system that is ready for use with *any* KG, within a reasonable amount of time (e.g., 24 hours), and without *any* training data? To achieve this, a model must efficiently and accurately learn to reason over a KG with no prior knowledge of the query distribution or the KG semantics.

BYOKG takes inspiration from the human tendency to be curious—seeking challenges and developing knowledge even in the absence of well-defined rewards (Oudeyer et al., 2016; Di Domenico and Ryan, 2017). Given a new KG, a human practitioner begins familiarizing themselves with the graph by inspecting random nodes and analyzing the various properties[1] found in the node neighborhoods. As this process continues (crucially, without a task-specific information need in mind), the practitioner develops an intuition for the set of questions that can be answered with the information present in the KG.

To mechanize this human tendency, BYOKG consists of an exploration module, which combines random walks over the KG nodes with a set of graph operations (e.g. `COUNT`, `ARGMAX`, `>=`, etc.) to produce programs of varying degrees of complexity (STAGE 1; fig. 1). Our explorer is symbolic in nature and has the goal of maximizing diversity within the generated programs, akin to curiosity-driven human learning (Ryan and Deci, 2000).

After sampling a diverse set of programs, BYOKG leverages the strong generalization abil-

[1] For e.g., `https://prop-explorer.toolforge.org/`.

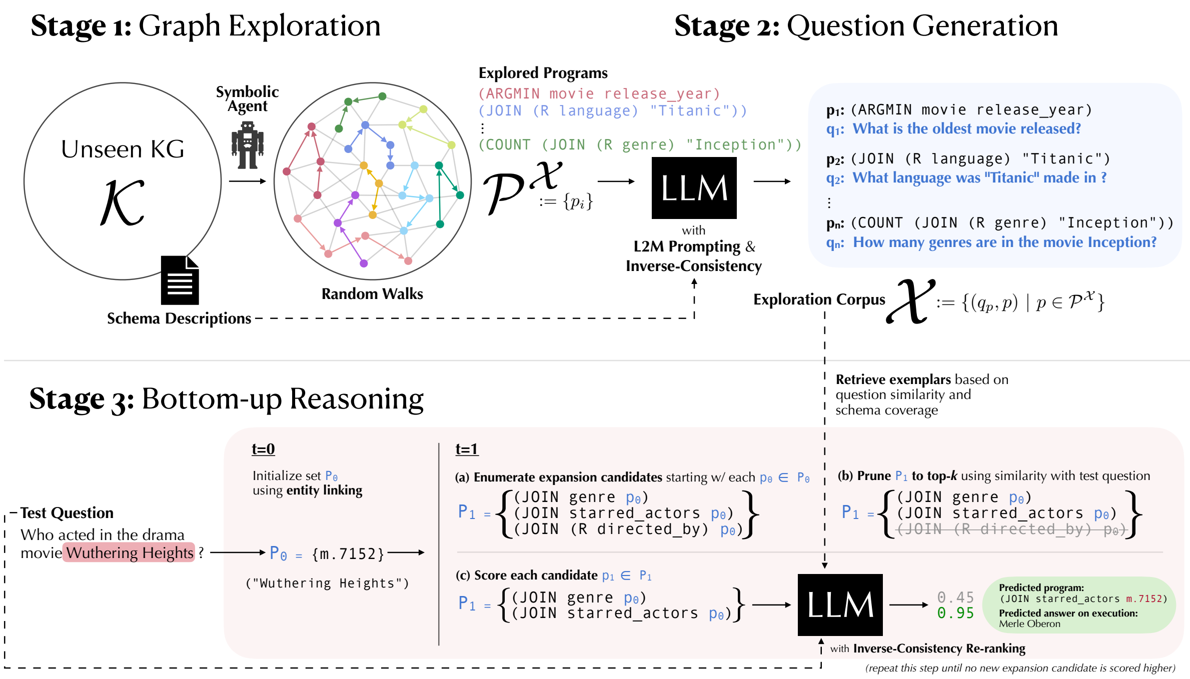

Figure 1: **Overview.** Given a new KG, a symbolic graph explorer generates diverse programs. Next, an LLM generates questions for the programs using descriptions of schema items, which are then stored in an exploration corpus. This process is done once for a KG. To answer a given question, BYOKG adopts a grounded reasoning approach that iteratively synthesizes the correct program using retrieved exemplars from the exploration corpus.

ity of large language models (LLMs) (Brown et al., 2020; Wei et al., 2022; Touvron et al., 2023) to generate questions for each program (STAGE 2). However, we find that LLM outputs are often semantically inaccurate with respect to the program, particularly in the zero-shot setting. To improve LLM generation, we, thus, develop a novel inverse-consistency re-ranking method, which computes scores for generated queries based on the likelihood of the query *re-generating* the program. We also incorporate least-to-most (L2M) prompting (Zhou et al., 2023) to improve generation for multi-hop programs. Empirically, we find that both techniques greatly improve the accuracy of question generation and are essential in allowing us to operate within our unsupervised setting.

Finally, BYOKG uses the explored query-program pairs to perform reasoning in order to answer user queries (STAGE 3). With the motivation of designing a QA system that can work on any KG, we opt for a semi-parametric approach instead of KG-specific fine-tuning. In particular, we build upon Pangu (Gu et al., 2023), an LLM-based discriminative procedure that iteratively synthesizes the predicted program guided by retrieved exemplars from the training data. We introduce several modifications, including a pruning step, which dramatically reduces runtime (by 88%) as well as increases accuracy.

In summary, our contributions are as follows—**(a)** we introduce **BYOKG**, a method that allows practitioners to "bring their own KG" with *no* training data and have a natural language query interface ready within a day. **(b)** Inspired by intrinsic motivation, we develop an LLM-backed exploration module, which explores the KG to gather query-program exemplars. We demonstrate that ICL-based models that use our exploration perform competitively with models that use annotated training data. **(c)** We show that our proposed inverse-consistency re-ranking and L2M prompting greatly improve the quality of zero-shot generation. **(d)** We demonstrate that BYOKG is effective on both small- (MoviesKG; 10^5 edges) and large-scale KGs (Freebase; 3×10^9 edges). On GrailQA and MetaQA, BYOKG provides dramatic improvements of 27.89 and 59.88 F1, respectively, over a zero-shot baseline. **(e)** We show that BYOKG scales with model size and even *outperforms* a state-of-the-art fine-tuned model on zero-shot queries by 7.08 F1 on GrailQA using a larger LM (GPT-3.5). **(f)** Finally, we demonstrate that BYOKG is able to operate in arbitrary domains without training data, showing a strong 46.33 F1 gain using a materials science KG.

2 Task Definition

KGQA. A knowledge graph $\mathcal{K}$ is a set of triples, or facts, of the form $\mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{C}$ denote entities, binary relations, literals, and classes (entity types), respectively. KGQA is then defined as the task of finding a set of answers $\mathcal{A}$ over graph $\mathcal{K}$ for a natural language question q. In program synthesis, the task is evaluated as mapping q to a program p_q (e.g. SPARQL or s-expression (Su et al., 2016)), which can deterministically be executed using a query engine to generate the answer set, i.e. $\texttt{eval}^{\mathcal{K}}(p_q) = \mathcal{A}_q$.

Unsupervised KGQA. We define *unsupervised* KGQA as a zero-shot setting where no query supervision over the target distribution is available[2]. Unsupervised KGQA jointly addresses multiple dimensions of generalization—linguistic variability (Khosla et al., 2023), query complexity (Keysers et al., 2020; Gu et al., 2021; Sen et al., 2023), domain transfer (Gu et al., 2021; Baek et al., 2023), and schema generalization (Das et al., 2021; Badenes-Olmedo and Corcho, 2023)—each of which has individually been shown to pose challenges to current QA systems.

3 Method

BYOKG consists of three stages—**graph exploration** (§3.1), **query generation** (§3.2), and **reasoning** (§3.3). First, our method explores the KG to enumerate a diverse set of executable programs. Next, each explored program is converted into a natural language question by prompting an LLM with schema descriptions of the relations and classes in the program. Finally, BYOKG leverages its acquired knowledge from exploration to ground a bottom-up inference procedure to iteratively generate the final program.

3.1 Symbolic Graph Exploration

The goal of graph exploration is to enumerate possible programs that may be queried at test time. However, exhaustive enumeration is often impractical with real-world KGs due to limited compute and time budgets. Instead, we construct a set of **explored programs** $\mathcal{P}^{\mathcal{X}}$ that provides approximate coverage of query patterns supported by the KG. BYOKG uses a symbolic, graph-based (Su et al., 2016) random walk procedure to enumerate a diverse set of executable programs.

Concretely, a symbolic agent begins exploration by initializing a sub-program p_0 at $t = 0$ with a class $c_0 \sim \mathcal{C}$. Next, the agent determines $S_{p_0} := \{s \mid s \in \mathcal{R} \cup \mathcal{C} : \texttt{reachable}(p_0, s)\}$, the set of schema items reachable from p_0. The agent then picks an item $s_0 \sim S_{p_0}$ to extend the sub-program into p_1. This process is repeated until the desired complexity of the program (i.e. relation count) is satisfied. The agent then, optionally, samples a program function $f \sim \mathcal{F}$ to apply over p_t, where $\mathcal{F}$ contains operators such as COUNT, comparatives, and superlatives. To encourage diversity, we discard p_t and repeat the process if $\mathcal{P}^{\mathcal{X}}$ already contains p_t[3]. Finally, we ground the classes appearing in p_t randomly by sampling from $\{e \mid e \in \mathcal{E} : \texttt{eval}^{\mathcal{K}}(p_t^e) \neq \emptyset\}$, the set of entities that lead to non-empty answer sets on program execution. The grounded p_t^e is then added to $\mathcal{P}^{\mathcal{X}}$.

3.2 Natural Language Query Generation

For each $p \in \mathcal{P}^{\mathcal{X}}$, we next generate a natural language question q_p to build an **exploration corpus** $\mathcal{X} := \{(q_p, p) \mid p \in \mathcal{P}^{\mathcal{X}}\}$ of query-program pairs. To generate questions, we prompt an LLM with instructions and textual descriptions of schema items relevant to each program (see A.5). Generating accurate output without in-context exemplars, however, is challenging. To elicit reliable zero-shot generation, we, therefore, utilize two techniques—(1) least-to-most prompting (Zhou et al., 2023), which generates outputs for complex programs in a step-by-step manner, and (2) a novel *inverse-consistency* method to re-rank LLM generations by scoring the inverse task of program generation.

3.2.1 Least-to-Most Prompting

Several prior works (Jung et al., 2022; Zhou et al., 2023; Drozdov et al., 2023) have tackled complex generative tasks by providing intermediate supervision via iteratively prompting the model with its own generations as additional context. Using these observations, we implement a least-to-most (L2M) prompting strategy that first decomposes p into simpler sub-programs $(p^1, p^2, \ldots, p^n)$ of increasing complexity using bottom-up parsing. We then generate a question q_{p^i} for each sub-program, appending each (p^j, q_{p^j}) for $j < i$ as additional

[2] This is a stronger generalization requirement than prior work (Gu et al., 2021), where queries with even a single schema item unseen at training are considered zero-shot.

[3] We set the max. number of programs per pattern to 5.

demonstrations in the prompt (see E.1).[4] In A.4, we show that L2M is crucial in unlocking deliberate, "System 2" reasoning (Kahneman, 2011) for complex queries in the zero-shot setting.

3.2.2 Inverse-Consistency Re-ranking

We observe that even when LLMs *can* produce the right answer within a top-k set of generations (e.g., from beam search), they do not always rank the correct answer as the top prediction, particularly with smaller models and in the unsupervised setting, rendering their use infeasible (see F.1). To tackle this, we introduce a re-ranking mechanism that scores output sequences from an LM using the likelihood of an *inverse* task, i.e. how likely the *input* sequence is given the output.

Concretely, consider a generative task $T := y \mid I, D, x$, where x is a sequence of query tokens, y is the target sequence of tokens to be predicted by a decoding algorithm, I is the textual instruction for the task, and D is the set of in-context demonstrations ($D = \emptyset$ in the unsupervised setting). The prediction y_{pred} for T is the top-ranked sequence from a list of candidates $\mathbf{y}_{\text{cands}}$ generated by the decoding algorithm measured using length-normalized log-probability scores, i.e. $y_{\text{pred}} := \arg\max_{y \in \mathbf{y}_{\text{cands}}} \log \Pr(y \mid I, D, x) \ / \ |y|$. To re-rank $\mathbf{y}_{\text{cands}}$, we now construct the following *inverse* task:

$$T^{-1} := x \mid I^{-1}, D^{-1}, y,$$

i.e. the task of predicting the query sequence x given an output sequence y from T, along with a new instruction I^{-1} for the inverse task and, optionally, an inverted demonstration set D^{-1}. For e.g., for the task of query generation, the inverse task is program synthesis. The new prediction is then given by

$$y_{\text{pred}} := \arg\max_{y \in \mathbf{y}_{\text{cands}}} \log \Pr(x \mid I^{-1}, D^{-1}, y) \ / \ |x|.$$

Scoring T^{-1} for a single y requires only one forward pass to get the next-token logit distribution at each position, allowing efficient computation of the log-probability score of the fixed-sequence x given y. Scores over the entire set $\mathbf{y}_{\text{cands}}$ can simply be computed using a batched forward pass. Inverse-consistency indeed improves generation accuracy (A.3) and enables BYOKG to use smaller models to scale exploration. We also note the close relation with PMI-scoring (Holtzman et al., 2021), but observe differing behavior in practice (see A.8).

3.3 Bottom-up Reasoning

With a corpus of query-program pairs in place, we now require a method to synthesize programs given natural language queries at test time. To use a single model with *any* KG, a key desiderata is to avoid KG-specific parameter tuning (Khosla et al., 2023). We, therefore, use an ICL approach using demonstrations from the exploration corpus within an enumerate-and-rank procedure. We adapt the method in Gu et al. (2023) with modifications that provide speed and accuracy gains to allow BYOKG to operate well in the unsupervised setting.

Concretely, given a test question q_{test}, BYOKG first instantiates a set of candidate sub-programs P_0 at $t = 0$ with all the topic entities, classes, and literals found in the question, extracted using off-the-shelf linkers (Li et al., 2020; Agarwal et al., 2022). In each subsequent timestep t, the reasoner determines which sub-programs from the previous step should further be extended. To do this, we use an LLM to compute[5] the likelihood of each sub-program being the parse for q_{test} conditioned on retrieved demonstrations D_{test} from exploration, and retain the top-k candidates

$$P_{t-1} := \underset{p^i_{t-1} \in P_{t-1}}{\arg\operatorname{topk}} \texttt{LLM}(p^i_{t-1}, q_{\text{test}}, D_{\text{test}}).$$

We additionally define

$$P_{\text{best}} := \underset{p \in P_{\text{best}} \cup P_{t-1}}{\arg\operatorname{topk}} \texttt{LLM}(p, q_{\text{test}}, D_{\text{test}})$$

as the best set of candidates across timesteps. After scoring, the reasoner extends each $p^i_{t-1} \in P_{t-1}$ using an extensible set of program expansion heuristics (Gu and Su, 2022) to construct the candidate set for the next timestep,

$$P_t := \left\{ \texttt{extend}(p^i_{t-1}, S_{p^i_{t-1}}, P_{\text{best}}) \,\middle|\, p^i_{t-1} \in P_{t-1} \right\},$$

where $S_{p^i_{t-1}}$ is the set of schema items reachable from p^i_{t-1} and P_{best} is the set of best-k candidates so far. The process terminates when no new sub-program is added to P_{best}, at which point we output the prediction $p_{\text{pred}} := \arg\max_{p \in P_{\text{best}}} \texttt{LLM}(p, q_{\text{test}}, D_{\text{test}})$.

[4] Query decomposition with s-expressions is straightforward—starting from the inner-most clause, the next sub-program is generated by simply including all the terms within the next parenthetic level.

[5] LLM scoring tends to prefer candidates with repeated relations. We, thus, penalize the final score based on the count of repeated relations. We do not add this penalty on MoviesKG due to the formulaic nature of the evaluation set.

ICL from exploration. To make predictions using an LLM, BYOKG takes a few-shot prompting approach to score candidate sub-programs conditioned on reasoning patterns for similar questions seen during exploration. A typical approach is to retrieve the k-most similar exemplars from $\mathcal{X}$ using the cosine similarity of exploration queries with the test query as measured using a sentence embedding model (Reimers and Gurevych, 2019). Following prior work (Thai et al., 2023), we additionally anonymize topic entities mentioned within questions to retrieve similar program *patterns* instead of similar topic entities. For instance, the question *"How many trophies has Manchester United won?"* would be anonymized to *"How many trophies has sports.team won?"*.

Candidate pruning. Scoring candidates can entail arbitrary latency depending on the number of candidates to score, making reasoning impractically slow when the candidate set P_t to be scored is very large (Table 10). We, therefore, introduce a candidate pruning step that restricts the size of the candidate set to at most 10 at each step of reasoning based on the similarity of anonymized candidate programs with the anonymized natural language test question using the sentence embedding model from retrieval. To keep our setup KG-agnostic, we do not fine-tune this model. As shown in A.6, we find that not only does pruning improve efficiency, but it also results in more accurate reasoning.

Inverse-consistency for candidate re-ranking. When schema items are *completely* unseen during exploration, we find that LLM scoring erroneously assigns high scores to irrelevant candidates that may resemble the retrieved exemplars (see F.2).

To address this problem, we re-use inverse-consistency (§3.2.2) to *re-rank* the final candidate set P_{best}. Concretely, we construct the inverse task, denoted by $\texttt{LLM}^{-1}(\cdot,\cdot)$, to be one of zero-shot question generation. To make predictions, we use a weighted combination of the original and inverse scores using weight α[6], resulting in

$$\texttt{rerank}(p,q,D) := \alpha\texttt{LLM}(p,q,D) \ + \backslash \\ (1-\alpha)\texttt{LLM}^{-1}(p,q),$$

which leads to the final prediction

$$p_{\text{pred}} := \underset{p\in P_{\text{best}}}{\arg\max}\ \texttt{rerank}(p, q_{\text{test}}, D_{\text{test}}).$$

[6]We do not tune α, in keeping with our setting of not assuming a dev set, and set its value to 0.5 in all experiments.

4 Experiments

4.1 Graphs and Datasets

For our larger-scale experiments, we use **Freebase** (Bollacker et al., 2008) and evaluate QA performance on the **GrailQA** (Gu et al., 2021) dataset. For smaller, domain-specific evaluation, we use **MoviesKG** (Miller et al., 2016) and the **MetaQA** (Zhang et al., 2018) dataset. Note that in the unsupervised setting, all datasets are o.o.d..[7]

4.2 Evaluation Metrics

Our primary metric is the **F1-score** between the predicted and reference answer sets. Several prior works (on MetaQA), however, only provide ranked entities. To compare, we report **Hits@1**, assigning rank 1 to each answer in our prediction set.

4.3 Models

We use **MPT-Instruct** (MosaicML-NLP-Team, 2023) (7B) for our main experiments. To demonstrate the scaling behavior of BYOKG, we additionally use **MPT-30B** as well as **GPT-3.5** (Brown et al., 2020) with the *text-davinci-003* variant[8].

4.4 Experimental Settings

4.4.1 Unsupervised

Our main experimental setting evaluates models with no access to *any* query supervision.

Zero-shot represents our bottom-up reasoning procedure from §3.3 but without any in-context demonstrations to score sub-programs at each step.

ICL + Exploration represents our proposed BYOKG method. In this setting, in-context demonstrations are retrieved from the exploration corpus $\mathcal{X}$, which we limit to 10K programs based on our time and compute budget. We also include in this setting results with Pangu-ICL (Gu et al., 2023), the few-shot variant of a KGQA method closely related to the bottom-up reasoning procedure of BYOKG.

4.4.2 Supervised

To situate our evaluations in the unsupervised setting, we also include a comparison with methods that have access to curated training data.

ICL + Train Set is the setting where both BYOKG and Pangu retrieve demonstrations from

[7]See Appendix C.1 for details on the datasets and KGs.

[8]Of the available variants, only *text-davinci*, *text-curie*, and *text-babbage* are compatible with BYOKG since we require access to log-probabilities to score sequences.

	Method	Model	Overall	I.I.D.	Compositional	Zero-shot
Supervised (w/ train set)	Pangu-FT (SOTA)	T5-3B	**81.7**	**88.8**	**81.5**	**78.5**
	Pangu-ICL + $\mathcal{T}_{1k}$	Codex	65.0	73.7	64.9	61.1
	Pangu-ICL† + $\mathcal{T}_{10k}$	MPT-7B	44.67	58.15	40.90	40.15
	BYOKG + $\mathcal{T}_{10k}$	MPT-7B	46.61	58.29	45.14	41.89
Unsupervised	Zero-shot	MPT-7B	18.58	19.13	16.34	19.33
	Pangu-ICL† + $\mathcal{X}$	MPT-7B	42.44 (Δ+23.86)	45.08	38.79	42.85
	BYOKG + $\mathcal{X}$ (OURS)	MPT-7B	**46.47** (Δ+27.89)	**48.91**	**43.22**	**46.80**

Table 1: **KGQA Results on GrailQA.** F1-scores for BYOKG in the unsupervised setting on the GrailQA test set compared to a zero-shot baseline and Pangu. For reference, we also report performance with models that use training data—ICL with randomly sampled training exemplars ($\mathcal{T}_{1k}$ and $\mathcal{T}_{10k}$) as well as a state-of-the-art fine-tuned model. We find that BYOKG + $\mathcal{X}$ improves zero-shot performance by 2.5x (nearly matching the performance of its supervised counterpart). BYOKG also demonstrates stable performance across generalization splits ($\sigma = 2.35$), whereas supervised methods ($\sigma = 7.09$) show drops in performance on the compositional and zero-shot splits. († indicates our re-implementaton)

Model	Overall	I.I.D.	Comp.	Z-shot
Pangu-FT	**81.68**	**92.81**	**79.97**	73.91
Pangu-Codex	65.0	73.7	64.9	61.1
BYOKG + $\mathcal{X}$ (OURS)				
MPT-7B	66.79	70.40	61.35	69.08
MPT-30B	69.58 (Δ+2.79)	73.10	65.14	70.95
GPT-3.5	75.16 (Δ+8.37)	73.89	70.33	**80.99**

Table 2: **BYOKG Accuracy v/s Model Scale.** F1-scores for BYOKG + $\mathcal{X}$ using 300 randomly sampled questions from the GrailQA dev set. (a) BYOKG shows gains in accuracy with improvements in the underlying LLM. (b) BYOKG with GPT-3.5 shows stable performance across generalization splits (unlike Pangu with training data). (c) BYOKG *outperforms* Pangu-FT on the zero-shot split by 7.08 points. (**Note:* Pangu-Codex test set results are included only to provide an estimate of ICL performance with a similar model.)

a randomly sampled subset of 10K training exemplars $\mathcal{T}_{10k}$. On GrailQA, we also report published Pangu-ICL (1000-shot) results with OpenAI Codex (Chen et al., 2021)[9].

Fine-tuned includes Pangu-FT, a fine-tuned T5-3B (Raffel et al., 2020) variant of Pangu trained using the full train set of 44K exemplars on GrailQA, and is currently the state-of-the-art (without ensembling). On MetaQA, we include NSM-FT (He et al., 2021), a fine-tuned method trained using teacher-student networks over 329K training exemplars. Although these models comprise dataset-specific parameters, we include them to provide an estimate of an upper-bound[10].

[9]LLM for instruction-following on code (now deprecated).

[10]No strict bound exists for unsupervised performance to be lower than supervised. See Fig. 2 for scaling trends.

5 Results

Exploration leads to substantial gains in the unsupervised setting. On both GrailQA with the Freebase KG (Table 1) and MetaQA with the MoviesKG (Table 3), we find that unsupervised exploration leads to dramatic gains over the zero-shot baseline. Specifically, our proposed BYOKG + $\mathcal{X}$ results in large 27.89 F1 (2.5x) and 59.88 F1 (4.9x) improvements on GrailQA and MetaQA, respectively.

BYOKG exhibits better compositional generalization than Pangu. On GrailQA, BYOKG outperforms Pangu by 4.03 F1 (Table 1) and on MetaQA by a large 20.63 F1 (Table 3) when evaluated with our exploration corpus. Note that exploration provides only partial coverage over evaluation queries (as shown in Table 6). Therefore, models must compositionally assemble sub-expressions from relevant exemplars to make predictions. For instance, on MetaQA, we find that training data provides perfect test pattern coverage, which translates to similar performance with both BYOKG and Pangu. With the exploration corpus, however, coverage of test patterns drops to nearly 70%, resulting in a large 30.93 point drop using Pangu and only 6.79 with BYOKG, highlighting the strong compositional generalizability of our method.

BYOKG with exploration is competitive with supervised ICL. We observe that BYOKG + $\mathcal{X}$ is able to nearly match BYOKG + $\mathcal{T}_{10k}$ (row 4 and 7 in Table 1) on GrailQA. Notably, we find that unsupervised BYOKG is, in fact, able to *outperform* supervised Pangu when the underlying base model is held constant (MPT-7B). On MetaQA, the

	Method	Overall		1-hop		2-hop		3-hop	
		F1	Hits@1	F1	Hits@1	F1	Hits@1	F1	Hits@1
Supervised (w/ train set)	NSM-FT (SOTA)	-	**98.82**	-	97.1	-	**99.9**	-	**98.9**
	Pangu-ICL† + $\mathcal{T}_{10k}$	**85.61**	92.38	97.88	**98.80**	**93.43**	94.21	**69.82**	86.01
	BYOKG + $\mathcal{T}_{10k}$	82.10	87.31	**97.95**	98.27	90.24	90.76	62.57	76.08
Unsupervised	Zero-shot	15.43	25.11	34.07	41.67	8.10	11.42	10.09	27.84
	Pangu-ICL† + $\mathcal{X}$	54.68 (Δ+39.25)	64.87	59.32	63.40	62.67	66.74	44.60	63.96
	BYOKG + $\mathcal{X}$ (OURS)	**75.31** (Δ+59.88)	**83.01**	**94.83**	**95.25**	**80.28**	**81.85**	**56.54**	**75.69**

Table 3: **KGQA Results on MetaQA.** F1-scores for BYOKG in the unsupervised setting on the MetaQA test set compared to a zero-shot baseline and Pangu. For reference, we also report supervised ICL baselines with 10K randomly sampled training examples ($\mathcal{T}_{10k}$) and NSM, a state-of-the-art fine-tuned LSTM. Exploration ($\mathcal{X}$) improves zero-shot F1 performance by 3.5x using Pangu and 4.9x using BYOKG. Further, BYOKG + $\mathcal{X}$ closes the gap with the best-performing supervised baseline to within only 10.3 F1. († indicates our re-implementaton; all ICL methods are evaluated using MPT-7B.)

gap between BYOKG + $\mathcal{X}$ and supervised ICL is a larger 6.79 F1, which can be explained by the formulaic nature of questions in MetaQA, resulting in all patterns being covered by the training set (see Table 6). Overall, our results demonstrate that exploration is a viable means to provide unsupervised grounding for reasoning.

BYOKG with exploration leads to more consistent performance across generalization splits versus supervised methods. In Table 1, we find that BYOKG + $\mathcal{X}$ demonstrates low variance (2.35 versus 7.09 standard deviation using $\mathcal{X}$ and $\mathcal{T}_{10k}$, respectively) in performance across generalization splits while methods using training data show fluctuations (drops) in performance on both compositional and zero-shot splits. We argue that the unsupervised nature of exploration allows BYOKG to discover reasoning patterns without additional bias introduced by a training distribution, thus allowing it to generalize well.

BYOKG improves with model scale. To evaluate potential gains with BYOKG by improving the underlying LLM, we compare KGQA performance using MPT-7B versus MPT-30B and GPT-3.5, a state-of-the-art instruction-tuned LLM from OpenAI. Due to a limited budget of \$100, we sample a small set of 300 questions from the GrailQA dev set and evaluate BYOKG + $\mathcal{X}$. Table 2 shows that improving the base model indeed leads to consistent gains in KGQA performance, with MPT-30B and GPT-3.5 showing improvements of 2.79 and 8.37 F1, respectively. BYOKG + GPT-3.5 additionally demonstrates more consistent performance across generalization splits as compared to Pangu-FT (state-of-the-art) and, notably, *outperforms* it on zero-shot queries by 7.08 F1.

Method	Overall	I.I.D.	Zero-shot
Zero-shot	15.92	13.75	22.42
BYOKG + $\mathcal{X}$	**62.25** (Δ+46.33)	**63.85**	**57.44**

Table 4: **KGQA Results on MatKG.** F1-scores for BYOKG with 9,445 explored programs on a test set of 100 questions (75/25 i.i.d./zero-shot) compared to a zero-shot baseline using MPT-7B.

Case Study: Materials Science KG. We, next, evaluate the ability of BYOKG to work in arbitrary, specialized domains by creating a natural language interface for an unseen KG from materials science using **MatKG** (Venugopal et al., 2022)[11]. Since the graph is not accompanied by a set of natural language questions, we randomly sample 100 programs up to 3-hops with unique query patterns and manually annotate them to construct a test set (see B). As shown in Table 4, BYOKG + $\mathcal{X}$ (with $|\mathcal{X}| \approx$ 10K) results in a large 46.33 F1 gain over zero-shot reasoning that uses no exploration.

Analyses and ablations. (a) As show in Appendix A.1, it is impractical to exhaustively sample all program patterns from real-world KGs when operating under a time budget, resulting in lower coverage with exploration compared with a curated training set (Δ-36.94, GrailQA; Δ-30.61, MetaQA). Despite this lower coverage, the competitive performance of BYOKG (Tables 1 and 3) points to its strong ability to reason with un-

[11]To the best of our knowledge, this KG is not part of the pre-training corpus for the MPT family of models. See https://www.mosaicml.com/blog/mpt-7b#building-with-mosaicml-platform.

seen patterns. **(b)** In Appendix A.2, we show that BYOKG continues to scale with additional exploration, notably showing a positive slope even at 44K programs[12]. Additionally, we find that inverse-consistency re-ranking allows BYOKG to match (and exceed) the performance of standard predictions with a nearly 9x reduction in exploration cost. **(c)** In Appendix A.3 and Appendix A.4, we verify the efficacy of inverse-consistency re-ranking and L2M for question generation. On human evaluations, we find that inverse-consistency provides a large 22.5 point gain in semantic accuracy and L2M results in a gain of 17.5 points. Additionally, we include an ablation in Appendix A.7 to show that inverse-consistency also improves reasoning accuracy (Δ+4.94 and Δ+0.83 F1 on GrailQA and MetaQA, respectively). **(d)** In Appendix A.5, we provide an ablation to verify the beneficial effect of providing natural language schema descriptions to the LLM for question generation. **(e)** Finally, in Appendix A.6, we analyze the effect of candidate pruning during reasoning and find that our most aggressive setting ($k = 10$) not only reduces inference cost/query to 13s (8x $\downarrow$ v/s no pruning) but also results in greater accuracy (Δ+2.5 F1).

6 Related Work[13]

KGQA Generalization. KGQA beyond i.i.d. samples has seen progress both in terms of new benchmarks (Gu et al., 2021; Dutt et al., 2023b) as well as methods (Yu et al., 2023; Shu et al., 2022a; Ye et al., 2022; Gu and Su, 2022). Recently, works have also investigated generalization to unseen KGs (Dutt et al., 2022; Gao et al., 2023). However, these methods all assume access to *some* curated training data, which is completely unavailable in our unsupervised setting. We also highlight Bio-SODA (Sima et al., 2021), which shares our unsupervised setting. Their approach uses string similarity to match query tokens with KG schema items, rank them using a PageRank-based importance measure, construct a query graph using Steiner trees, and finally convert the graphs into SPARQL queries. However, this method is unable to handle complex queries — aggregations, superlatives, comparatives, conjunctions, amongst others. In concurrent work, Li et al. (2023b) propose a method to train KGQA models from synthetic data using LLMs. Unlike BYOKG, however, their work utilizes unlabeled queries from the train set as weak supervision and is, thus, not fully unsupervised. Beyond structured queries, our work is also related to PAQ (Lewis et al., 2021), which over-generates questions over Wikipedia but, crucially, returns only a cached response at test time instead of reasoning as in BYOKG.

KGQA with ICL. Many recent works have attempted to unify LLMs and knowledge graphs (Tian et al., 2023; Tan et al., 2023; Li et al., 2023a). However, most prior works require a training corpus to retrieve in-context demonstrations, which is unavailable in our setting. A prior work that does operate in a completely zero-shot setting is Baek et al. (2023), where triples are retrieved from the KG to generate the final answer. However, this method does not provide the answer text alone due to a generative strategy[14] making it largely incomparable with BYOKG.

Grounded Multi-Step Reasoning. Bottom-up parsing iteratively builds a solution for complex problems in several prior works in semantic parsing (Rubin and Berant, 2021; Gu and Su, 2022; Ye et al., 2022; Gu et al., 2023). BYOKG further grounds each step of bottom-up parsing to the KG using a case-based reasoning (CBR) approach, which has widely been applied in various tasks, such as link prediction (Das et al., 2022), semantic parsing (Das et al., 2021; Awasthi et al., 2023), and reading comprehension (Thai et al., 2023).

7 Conclusion

We introduce **BYOKG**—a universal KGQA system to work with *any* target KG and without *any* human-annotated training data. BYOKG mimics curiosity-driven learning in humans by first exploring the unseen KG, followed by using the acquired knowledge to answer questions. Our method combines LLMs with graph traversal to explore the KG and then reason over the explored paths to answer arbitrary user queries over the graph. We further introduce techniques to improve zero-shot performance with LLMs, including an inverse-consistency re-ranking method. On two popular datasets and KGs, we demonstrate the efficacy of BYOKG and present detailed analyses of the several design choices.

[12]We set our maximum budget to 44K to mirror the size of the curated training set.

[13]Please refer to Appendix D for further related work.

[14]They use a "generative accuracy" metric, which considers a prediction correct if the tokens of an answer entity are found anywhere within the generated text.

Limitations

While BYOKG satisfies several desiderata that we set out to meet, we discuss a few limitations of our current system, which may serve as useful future directions for improvement.

Despite efforts to control generations from LLMs, we observe that BYOKG is susceptible to hallucinations. For instance, during question generation, models may generate semantically inaccurate queries for their corresponding programs. While BYOKG is robust to *some* noise during exploration, inaccuracies at scale are crippling to the retrieval-augmented reasoning procedure, which relies on coherent exemplars to score program candidates. A plausible explanation for this behavior is our restriction on using off-the-shelf pre-trained models that are not explicitly trained for KGQA. Future directions may explore using models pre-trained for KGQA or even KG-specific parameter tuning.

Second, while we reduce latency by 8x compared to a naive implementation by introducing candidate pruning, our iterative "System 2" reasoning may not satisfy stringent response time requirements, which are better served by single-shot inference. Caching does address this limitation to an extent, but future work may explore how programs can be synthesized more efficiently for complex, multi-hop queries.

Third, the primary goal of BYOKG is to provide a query interface without any human intervention. However, as a prerequisite, we assume the availability of a schema enumerating the classes and relations present in the KG along with their natural language descriptions. Our assumption is based on the common availability of such a file accompanying most real-world KGs. In the absence of this data, BYOKG would, thus, currently require human annotations. Further leaning on the broad-spectrum generalization abilities of LLMs, future work may explore automatically generating such schema descriptions.

Lastly, we find that while inverse-consistency does reliably improve the quality of both question generation as well candidate scoring during reasoning in aggregate, it can result in converting a previously correct vanilla prediction into an error. To further improve the accuracy of BYOKG, it would thus be a promising direction to determine when and when not to use inverse-consistency re-ranking.

Broader Impact

The methodology we use in BYOKG has the potential to improve information access in several domains that contain structured information but may lack the expertise or resources to construct complex query interfaces, dramatically improving the availability of information in previously opaque settings. This democratization of access, particularly in the public domain, also holds promise to empower regular individuals to be better-informed about policies and activities that directly affect them and may, consequently, improve their participation to build more egalitarian societies.

However, we caution that systems that build upon large language models should be deployed in real-world settings with utmost care. In particular, due to a growing trend of closed-source development and release of models, unaccompanied by refereed technical documents, it is not always apparent as to the nature of the data that LLMs are pre-trained on. This has the potential of perpetuating factual inaccuracies and biases prevalent in corpora collected from the internet. Indeed, BYOKG is not immune to these pathologies and future work should study and address methods to detect and prevent such behavior.

References

Dhruv Agarwal, Rico Angell, Nicholas Monath, and Andrew McCallum. 2022. Entity linking via explicit mention-mention coreference modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4644–4658, Seattle, United States. Association for Computational Linguistics.

Abhijeet Awasthi, Soumen Chakrabarti, and Sunita Sarawagi. 2023. Structured case-based reasoning for inference-time adaptation of text-to-sql parsers. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press.

Carlos Badenes-Olmedo and Oscar Corcho. 2023. Muheqa: Zero-shot question answering over multiple and heterogeneous knowledge bases. *Semantic Web*, (Preprint).

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Neurips*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Rajarshi Das. 2022. *Nonparametric Contextual Reasoning for Question Answering over Large Knowledge Bases*. Doctoral dissertation.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

Rajarshi Das, Ameya Godbole, Ankita Naik, Elliot Tower, Manzil Zaheer, Hannaneh Hajishirzi, Robin Jia, and Andrew McCallum. 2022. Knowledge base question answering by case-based reasoning over subgraphs. In *International conference on machine learning*, pages 4777–4793. PMLR.

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. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*.

Stefano I Di Domenico and Richard M Ryan. 2017. The emerging neuroscience of intrinsic motivation: A new frontier in self-determination research. *Frontiers in human neuroscience*.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*.

Ritam Dutt, Kasturi Bhattacharjee, Rashmi Gangadharaiah, Dan Roth, and Carolyn Rose. 2022. PerKGQA: Question answering over personalized knowledge graphs. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 253–268, Seattle, United States. Association for Computational Linguistics.

Ritam Dutt, Sopan Khosla, Vinayshekhar Bannihatti Kumar, and Rashmi Gangadharaiah. 2023a. Designing harder benchmarks for evaluating zero-shot generalizability in question answering over knowledge bases. In *ACL 2023 Workshop on Natural Language Reasoning and Structured Explanations*.

Ritam Dutt, Sopan Khosla, Vinayshekhar Bannihatti Kumar, and Rashmi Gangadharaiah. 2023b. Designing harder benchmarks for evaluating zero-shot generalizability in question answering over knowledge bases.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. 2023. Towards foundation models for knowledge graph reasoning. *arXiv preprint arXiv:2310.04562*.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*.

Jianfei Gao, Yangze Zhou, and Bruno Ribeiro. 2023. Double permutation equivariance for knowledge graph completion. *arXiv preprint arXiv:2302.01313*.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949, Toronto, Canada. 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 *Web Conference*.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. *arXiv preprint arXiv:2104.08315*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022b. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*.

Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. 2022. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1266–1279, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR*.

Sopan Khosla, Ritam Dutt, Vinayshekhar Bannihatti Kumar, and Rashmi Gangadharaiah. 2023. Exploring the reasons for non-generalizability of KBQA systems. In *The Fourth Workshop on Insights from Negative Results in NLP*, pages 88–93, Dubrovnik, Croatia. Association for Computational Linguistics.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. 2022. RankGen: Improving text generation with large ranking models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 199–232, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6433–6441, Online. Association for Computational Linguistics.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023a. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2023b. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *CACM*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799, Seattle, United States. Association for Computational Linguistics.

Alexander H Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

MosaicML-NLP-Team. 2023. Introducing mpt-30b: Raising the bar for open-source foundation models.

P-Y Oudeyer, Jacqueline Gottlieb, and Manuel Lopes. 2016. Intrinsic motivation, curiosity, and learning: Theory and applications in educational technologies. *Progress in brain research*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. 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.

Richard M Ryan and Edward L Deci. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67.

Devendra Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3781–3797, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Devendra Singh Sachan, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joelle Pineau, and Manzil Zaheer. 2023. Questions are all you need to train a dense passage retriever. *Transactions of the Association for Computational Linguistics*, 11:600–616.

Priyanka Sen, Sandeep Mavadia, and Amir Saffari. 2023. Knowledge graph-augmented language models for complex question answering. In *Workshop on NL Reasoning and Structured Explanation*.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022a. TIARA: Multi-grained retrieval for robust question answering over large knowledge base. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8108–8121, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022b. Tiara: Multi-grained retrieval for robust question answering over large knowledge bases. *arXiv preprint arXiv:2210.12925*.

Ana Claudia Sima, Tarcisio Mendes de Farias, Maria Anisimova, Christophe Dessimoz, Marc Robinson-Rechavi, Erich Zbinden, and Kurt Stockinger. 2021. Bio-soda: Enabling natural language question answering over knowledge graphs without training data. In *Proceedings of the 33rd International Conference on Scientific and Statistical Database Management*, SSDBM '21, page 61–72, New York, NY, USA. Association for Computing Machinery.

Ana Claudia Sima, Tarcisio Mendes de Farias, Maria Anisimova, Christophe Dessimoz, Marc Robinson-Rechavi, Erich Zbinden, and Kurt Stockinger. 2022. Bio-soda ux: enabling natural language question answering over knowledge graphs with user disambiguation. *Distributed and Parallel Databases*.

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.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL*.

Chuanyuan Tan, Yuehe Chen, Wenbiao Shao, and Wenliang Chen. 2023. Make a choice! knowledge base question answering with in-context learning. *arXiv preprint arXiv:2305.13972*.

Dung Thai, Dhruv Agarwal, Mudit Chaudhary, Rajarshi Das, Manzil Zaheer, Jay-Yoon Lee, Hannaneh Hajishirzi, and Andrew McCallum. 2023. Machine reading comprehension using case-based reasoning. *arXiv preprint arXiv:2305.14815*.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2023. Graph neural prompting with large language models. *arXiv preprint arXiv:2309.15427*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vineeth Venugopal, Sumit Pai, and Elsa Olivetti. 2022. The largest knowledge graph in materials science - entities, relations, and link prediction through graph representation learning. In *AI for Accelerated Materials Design NeurIPS 2022 Workshop*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. 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 1332–1342, Beijing, China. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *TMLR*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, Copenhagen, Denmark. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Manzil Zaheer, Kenneth Marino, Will Grathwohl, John Schultz, Wendy Shang, Sheila Babayan, Arun Ahuja, Ishita Dasgupta, Christine Kaeser-Chen, and Rob Fergus. 2022. Learning to navigate wikipedia by taking random walks. *Advances in Neural Information Processing Systems*, 35:1529–1541.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Appendices

We provide several supplementary details of our work and organize them as follows:

- Appendix A: **Analyses and Ablations**
- Appendix B: **MatKG Dataset**
- Appendix C: **Implementation Details**
- Appendix D: **Related Work**
- Appendix E: **Language Model Prompts**
- Appendix F: **Qualitative Examples**

A Appendix: Analyses and Ablations

In this section, we present a detailed analysis of the design choices made in BYOKG and how they affect downstream QA performance.

A.1 KG and Query Coverage with Exploration

Exploration statistics. Table 5 shows the results of unsupervised KG exploration on Freebase (Commons) as well as MoviesKG, including the distribution of programs of different complexity as well as the wall-clock time taken for the procedure. While program generation is inexpensive, the cost of question generation restricts the number of programs we can explore. We stop at 10K to meet our stated goal of readying a QA system within a day.

(budget of 10k programs)	**Freebase**	**MoviesKG**
Programs	10,000	10,000
1-hop	6,933	222
2-hop	2,589	1,779
3-hop	426	4,290
4-hop	52	3,709
Relations	4,178	18
Classes	1,681	7
Patterns	7,193	3,658
Sub-expressions	7,741	71
Time		
Exploration (mins)	46.5	24.4
Query Generation (hours)	10.4	24.0

Table 5: **Exploration Statistics** on Freebase and MoviesKG for a budget of 10K programs (capped at 5 programs per query pattern) using 3 Amazon EC2 p3dn.24xlarge machines. (*Note:* relation counts listed also include reverse relations.)

	GrailQA		**MetaQA**	
(in dev set)	$\mathcal{T}$	$\mathcal{X}$	$\mathcal{T}$	$\mathcal{X}$
Relations	82.49	76.89	100.00	100.00
Classes	85.43	91.56	100.00	100.00
Patterns	70.93	13.94	100.00	69.39
Sub-expressions	79.24	49.43	100.00	100.00

Table 6: **Distribution Coverage with Exploration** ($\mathcal{X}$) versus the full training data ($\mathcal{T}$) for queries in the development sets. On MetaQA, $\mathcal{X}$ provides high coverage (though nearly 30 points below $\mathcal{T}$ on query pattern coverage) due to the small size of MoviesKG. On GrailQA, with the larger Freebase KG, $\mathcal{X}$ shows a huge 56.99 points drop in query pattern coverage as well as a 29.81 drop for sub-expressions, leading to several queries being zero-shot versus when using the training data.

Distribution coverage. To effectively ground reasoning in BYOKG, exploration must be able to provide sufficient coverage over the queries being evaluated. We analyze how well our random exploration strategy with a budget of 10K performs compared to a curated training set in providing coverage over the evaluation distribution. Table 6 shows our results for coverage over relations, classes, program patterns, and sub-expressions (e.g. "`(COUNT #var)`", "`(ARGMIN type.datetime #var)`") found in the gold logical programs from the dev sets of GrailQA and MetaQA.

On MetaQA, we find that while exploration *can* find all schema items and sub-expressions, it misses nearly 30% of program patterns in the test distribution while the training set has perfect coverage. On GrailQA, both sub-expression and pattern coverage are much lower than training, with $\mathcal{X}$ observing 5x fewer test patterns and 1.6x fewer test sub-expressions than the training data. These gaps explain the difference in performance between supervised methods and BYOKG + $\mathcal{X}$, which is completely zero-shot (Table 1 and Table 3). This gap also highlights a future direction for improving BYOKG by incorporating more guidance into exploration that goes beyond diversity alone.

A.2 QA Accuracy v/s Exploration Budget

As shown in Table 6, real-world KGs, such as Freebase, are intractable to exhaustively explore resulting in only approximate coverage. Here, we evaluate the budget-accuracy trade-off of BYOKG, i.e. how the *amount* of exploration affects downstream QA performance. For this analysis, we randomly sub-sample multiple sets $\mathcal{X}_k$ of varying sizes k from $\mathcal{X}$, which we then

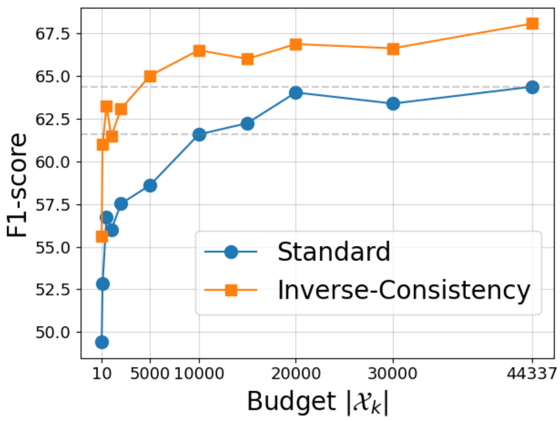

Figure 2: **Accuracy v/s Exploration Budget.** F1-scores with ByoKG + $\mathcal{X}_k$ using MPT-7B. ByoKG shows consistent gains with increasing exploration budget, notably showing a positive slope even at the maximum budget, indicating room for further improvement. Further, inverse-consistency candidate re-ranking improves performance at all budget levels and outperforms standard predictions at $\mathcal{X}_k = 10$K with only 500 programs (20x reduction) and $\mathcal{X}_k = 44$K with only 5K programs (9x reduction).

use to answer questions over a sub-sampled set of 3,000 questions (1k from each split) from the GrailQA dev set. In Fig 2, we plot F1-scores for ByoKG + $\mathcal{X}_k$. ByoKG shows steady improvements with more exploration, notably showing a positive slope even at 44K programs (our maximum due to budget constraints).

Inverse-consistency. Additionally, Fig. 2 shows that re-ranking improves performance at all budget levels. Notably, re-ranking recovers (and exceeds) the performance of standard predictions at the maximum budget with only a small set of 500 programs, i.e. a 20x reduction in exploration cost, which translates to a wall-clock setup time of only 1.6 hours (versus 1.3 *days* for 10K programs). Additionally, performance at the maximum budget of 44K programs can be matched using only 5K programs with inverse-consistency (9x reduction).

A.3 Inverse-Consistency for Question Generation

We evaluate the effect of inverse-consistency re-ranking on the quality of question generation. Table 7 shows a comparison between the top-1 generation from a standard beam-search procedure versus the inverse-consistency re-ranked output on 3,000 randomly sampled questions from the GrailQA dev set. We use three automatic generation metrics – ROUGE-1 (Lin, 2004), BLEU (Papineni et al., 2002), and BERTscore (Zhang* et al., 2020) – computed with respect to the human-annotated gold references in the dataset. Our results show that inverse-consistency indeed improves generation quality, as measured on all metrics. We further inspect 40 randomly sampled questions for semantic accuracy using both methods, and find inverse-consistency generates accurate output for 70% of questions, 22.5 points more than standard beam-search.

Metrics	Standard	Inverse-Consistency
ROUGE-1	48.17	**52.81** (Δ+4.64)
BLEU	31.54	**38.63** (Δ+7.09)
BERTscore	87.17	**88.33** (Δ+1.16)
Human Evaluation	47.50	**70.00** (Δ+22.50)

Table 7: **Inverse-Consistency for Question Generation.** Generation quality with inverse-consistency re-ranking compared with standard top-1 predictions from beam search using MPT-7B. Inverse-consistency improves generation quality as measured on both automatic and human evaluation metrics.

Model	Standard	Least-to-Most
MPT-7B	55.0	70.0
MPT-30B	60.0	80.0
Mean	57.5	**75.0** (Δ+17.5)

Table 8: **L2M Question Generation.** Human-evaluated semantic accuracy of question generation using L2M prompting versus standard single-shot generation over a random sample of 40 questions from the GrailQA dev set. L2M prompting improves accuracy of generated questions by a significant 17.5 points.

A.4 L2M for Question Generation

Here, we analyze the effect of L2M-prompting for question generation compared with standard, single-shot prompting. To conduct this analysis, we annotate a set of 40 questions and verify the semantic accuracy of the generated questions with respect to the corresponding logical programs. Table 8 shows our results, where we find that L2M prompting provides an 18.7 point improvement over standard decoding.

A.5 Schema Supervision for Question Generation

We evaluate the effect of providing natural language schema descriptions to the LLM during question generation. As shown in Table 9, we find that

	Standard	Schema
ROUGE-1	51.40	**52.81** (Δ+1.41)
BLEU	35.99	**38.63** (Δ+2.64)
BERTscore	87.59	**88.33** (Δ+0.74)

Table 9: **Schema Supervision for Question Generation.** Generation quality with schema descriptions injected into the prompt compared with standard prediction with only the query using MPT-7B over 3,000 randomly sampled questions from the GrailQA dev set.

schema supervision improves generation quality as measured by each automatic metric.

k	Answer-Recall	Answer-F1	Latency (sec/q)
∞ (Pangu)	**100.00**	59.70	110.1
50	98.67	**63.07**	20.2
20	95.33	62.95	15.1
10	84.67	62.20	**13.2**

Table 10: **Effect of Candidate Pruning.** Performance of BYOKG + $\mathcal{X}$ on a sub-sampled set of 300 questions from the GrailQA dev set at different pruning thresholds k for candidate set P_t. Answer-recall is the oracle recall of the gold program, answer-F1 measures KGQA performance, and latency is the average time per question over 300 questions. Evaluation is run with one Amazon EC2 p3dn.24xlarge machine using MPT-7B *without* inverse-consistency re-ranking and *without* caching. Aggressive pruning at $k = 10$ results in the most efficient reasoning with an accuracy gain of 2.5 F1 over no pruning.

A.6 Candidate Pruning for Reasoning

As noted in §3.3, we introduce candidate pruning in BYOKG in order to bound the latency at each reasoning step. This is in contrast to Pangu, which incurs high latency due to scoring every enumerated candidate. We analyze the effect of pruning in Table 10 on (1) the reachability of the gold program (answer-recall), (2) KGQA F1-scores, and (3) the latency per question[15]. With no pruning (Pangu), we encounter prohibitive runtimes of nearly 2 minutes per query, which is substantially reduced at $k = 10$ to 13s (8x speed-up). Surprisingly, we also find that aggressive pruning ($k = 10$) results in improved reasoning accuracy (+2.5 F1 v/s at $k = \infty$). In practice, we note that the latency of BYOKG will continue to improve as more queries are served due to caching results from SPARQL executions.

[15] In practice, we cache responses from the SPARQL engine to improve latency over time, but turn caching off for this evaluation. Also, $k = \infty$ refers to no pruning.

Dataset	Standard	Inverse-Consistency
GrailQA	61.58	**66.52** (Δ+4.94)
MetaQA	82.22	**83.05** (Δ+0.83)

Table 11: **Candidate Re-ranking with Inverse-Consistency.** F1-scores of BYOKG + $\mathcal{X}$ with inverse-consistency re-ranking compared to standard top-1 predictions over a sub-sampled set of 3K questions from the GrailQA dev set and the MetaQA test set. Inverse-consistency improves performance on both datasets.

A.7 Inverse-Consistency for Candidate Re-ranking

As described in §3.3, we find that inverse-consistency re-ranking during reasoning helps recover from errors where exploration does not provide coverage over the test questions. Table 11 shows a comparison of F1 accuracy with standard scoring v/s inverse-consistency re-ranked outputs. Re-ranked programs P_{best} are computed using $\texttt{rerank}(\cdot,\cdot)$ with $\alpha = 0.5$. We find that re-ranking provides a significant gain of 4.94 F1 on GrailQA, while MetaQA performance increases by 0.83. The modest gains on MetaQA, may be attributed to higher pattern and sub-expression coverage during exploration as compared to GrailQA (Table 6), resulting in fewer instances where re-ranking is required.

A.8 Inverse-Consistency v/s PMI

	Inverse-Consistency	PMI_{DC}
ROUGE-1	**52.71**	42.97 (Δ-9.74)
BLEU	**39.94**	23.52 (Δ-16.42)
BERTscore	**88.64**	85.78 (Δ-2.86)

Table 12: **Re-ranking with Inverse-Consistency v/s PMI_{DC} for Question Generation.** Generation quality as measured using automatic metrics using MPT-7B over 100 randomly sampled questions from the GrailQA dev set.

	Inverse-Consistency	PMI_{DC}
F1-score	**66.52**	65.02 (Δ-1.5)

Table 13: **Re-ranking with Inverse-Consistency v/s PMI_{DC} for Reasoning.** F1-scores using MPT-7B over 3K randomly sampled questions from the GrailQA dev set.

Holtzman et al. (2021) propose the domain-conditional pointwise mutual information (PMI_{DC}) scoring function, i.e. $\log \Pr(y|x) \ / \ \Pr(y|x_{\text{domain}})$

to address the “surface form competition” hypothesis, which aims to explain miscalibrated outputs from LLMs, resulting in low accuracy in zero-shot settings. While our inverse-consistency formulation $\log \Pr(x|y)$ should, in theory, provide the same ordering as PMI_{DC}, we evaluate how these methods compare as re-ranking techniques in practice. We run evaluations on sub-sampled examples from the GrailQA dev set for both question generation (Table 12) and candidate re-ranking during reasoning (Table 13). For question generation, we set x_{domain} to “`### English Question:\n`” and for reasoning, we set x_{domain} to “`### Logical Form:\n`”. We find that in practice the methods exhibit different behaviors, with inverse-consistency outperforming PMI_{DC} on both question generation and reasoning. A possible explanation for this variation is LLM sensitivity to the choice of prompt constructions to calculate the terms in the re-ranked expressions.

B Appendix: MatKG Dataset

Annotation Procedure. To evaluate BYOKG using MatKG, we annotate a set of 100 programs with natural language questions using 2 researchers from our team. In particular, we take our set of 10K explored programs and randomly sample 100 programs such that 75 programs are i.i.d. for the exploration set, while 25 are o.o.d. or unseen. We then randomly split the 100 questions into two sets and iteratively provide each annotator the sampled program text, natural language descriptions for the relations in the program, and natural language descriptions for the classes in the program. The annotator is then prompted to enter a natural language question based on this information. We release our annotated dataset for reproducibility and future research under the MIT License: https://drive.google.com/file/d/1o8CG9isSOScTZ3Ji1-71EzBEZoZqvnCR/view?usp=drive_link.

Annotation Examples. We provide a few examples from the annotated test set:

```
Program: (AND material (AND (JOIN material.
descriptor \"Bars\") (JOIN (R synthesis_method.
material) \"Ccs\")))
Query: which materials have been synthesized
using ccs and can be described as bars?

Program: (COUNT (AND descriptor (AND (JOIN (R
property.descriptor) \"Free Energy Diagram\") (
JOIN (R characterization_method.descriptor) \"
SEM Surface\"))))
Query: how many descriptors have property free
energy diagram and have characterization method
sem surface?

Program: (AND application (JOIN (R
characterization_method.application) (JOIN (R
property.characterization_method) \"Basalts\")))
Query: the characterization method of basalts
has what all applications?
```

C Appendix: Implementation Details

C.1 Graphs and Datasets

	Split	GrailQA	MetaQA	MatKG
$\|\mathcal{Q}\|$	Train	44,337	329,282	-
	Dev	6,763	39,138	-
	Test	13,231	39,093	100
$\|\mathcal{R}\|$	All	3,720	9	21
$\|\mathcal{C}\|$	All	1,534	7	7
$\|\mathcal{E}\|$	All	32,585	43,692	70,002

Table 14: **GrailQA, MetaQA, and MatKG Statistics.** Note that the relation counts do not include inverse relations.

Freebase (Bollacker et al., 2008) is a large-scale, open-domain KG containing over 100 domains, 45 million entities, and 3 billion facts. We use the **GrailQA** (Gu et al., 2021) dataset, which evaluates three levels of generalization—i.i.d., compositional (novel combinations of seen constructs), and zero-shot (unseen schema items)—and also features diverse questions of varying complexity (up to 4-hop) and aggregation functions (e.g. `COUNT` and comparatives). GrailQA was constructed with the help of 6,685 crowdworkers and restricts the KG to a high-quality Commons subset, which covers 86 unique domains.

MoviesKG is a small-scale, domain-specific KG provided by the WikiMovies dataset (Miller et al., 2016), containing triples that map movies to attributes such as actors, genres, and ratings. Unlike previous work, we convert the provided triples of entity labels into a structured store where entities with the same label name may be assigned different entity IDs if they represent unique concepts.[16] The accompanying dataset we use is **MetaQA** (Zhang et al., 2018), which consists of more than 400K multi-hop (up to 3-hop) questions.

[16]For e.g., “Jungle Book” may either refer to the 1967 or the 2016 movie, but would incorrectly be considered the same entity in past work. We will release a corrected set of triples and a new set of answers for MetaQA based on this change.

MatKG (Venugopal et al., 2022) represents the largest KG in the materials science and was automatically generated using LLMs from 4 million scientific papers resulting in 70K entities and 5.4 million unique triples, including chemistry, structure, property, application, synthesis, and characterization data as of our study (we use version 1.2 of the KG).

C.2 Models

MPT-Instruct (MosaicML-NLP-Team, 2023) is a decoder-style transformer pre-trained on 1T tokens of English text and code, followed by instruction fine-tuning on the Databricks-Dolly-15k (Conover et al., 2023) and Anthropic Helpful and Harmless datasets (Ganguli et al., 2022). We use the 7B model for our main experiments and also show a small-scale experiment with 30B to verify the efficacy of BYOKG at scale.

GPT-3.5 (Brown et al., 2020) is a state-of-the-art, closed-source model from OpenAI. We conduct a small-scale experiment (constrained by budget) using the *text-davinci-003* variant to demonstrate the scaling behaviors of BYOKG (§5).

C.3 Computing Infrastructure & Software

For compute, we have access to 3 Amazon EC2 p3dn.24xlarge instances (see `https://aws.amazon.com/ec2/instance-types/p3/`). Our experiments are run using PyTorch (Paszke et al., 2019) and utilize the Huggingface Transformers library (Wolf et al., 2020) and models hosted on Huggingface to access the LLMs in our work. For executing logical programs on the KG, we use the OpenLink Virtuoso SPARQL Engine using one of our EC2 machines (recommended RAM is 100G). While querying the SPARQL server, we limit each request to timeout after 5s. For further details about how to set up Virtuoso, we point to the following documentation: `https://github.com/dki-lab/Freebase-Setup/`. We additionally note that we will provide detailed instructions for our setup in the code repository that will be released publicly.

C.4 LLM Decoding Parameters

We use the following decoding arguments with the `generate()` call of HuggingFace's `AutoModelForCausalLM`:

```
default_decoding_args = {
    "max_new_tokens": 100,
    "do_sample": False, # enable sampling
    "top_p": 0.9, # nucleus sampling
    "temperature": 0.6, # lower makes the
    distribution sharper
    "min_length": None,
    "use_cache": True,
    "top_k": 100, # restrict to top-k probability
     tokens
    "repetition_penalty": 1., # 1 means no
    penalty; up to inf
    "length_penalty": 1., # length_penalty > 0.0
    == longer sequences; length_penalty < 0.0 ==
    shorter sequences
    "num_beams": 10, # beam search
    "num_return_sequences": 10, # number of beams
     to return
    "no_repeat_ngram_size": 10,
    "renormalize_logits": True
}
```

C.5 Reasoning Implementation

Program Expansion Heuristics. We re-implement the Freebase-based expansion heuristics detailed in Gu et al. (2023), to allow operating with arbitrary KGs that may then be setup with just a file of triples.

Entity Linking. For GrailQA, we utilize the entity linking results from Shu et al. (2022b) made available by Gu et al. (2023). For MetaQA, a simple string-matching approach results in perfect EL accuracy. For MatKG, we only evaluate with gold entity links, which are made available when automatically sampling programs.

D Appendix: Related Work

KGQA Generalization. Another line of work investigates pipelines for constructing semantic parsers for new KGs by generating training data automatically (Wang et al., 2015; Liang et al., 2016; Su et al., 2016; Gu et al., 2021). Each of these methods, however, includes a human annotation step to generate the final training data whereas BYOKG is able to operate without any supervision.

Galkin et al. (2023) recently introduced a foundational model to learn transferable representations for KGQA that allows them to generalize to unseen graphs without any training data. While similar in motivation to BYOKG, they do not handle natural language queries.

Planning and RL. Reasoning in BYOKG can be seen as iteratively constructing a plan to *navigate* the KG conditioned on a test query. Many prior works take a similar view and use reinforcement learning to construct path-finding algorithms for KGQA (Xiong et al., 2017; Das et al., 2018). These methods, however, were not designed to handle natural language queries. Several recent works also investigate the use of LMs as planners to navigate environments other than KGs, such as in robotics (Huang et al., 2022b,a), unstructured reasoning (Zaheer et al., 2022; Yao et al., 2023; Shinn et al., 2023), game environments (Wang et al., 2023), and web navigation (Deng et al., 2023).

LM Generation Re-ranking. Beyond LM decoding (Holtzman et al., 2020; Lu et al., 2022), recent work has also studied how to best *rank* sequences generated by LMs. For instance, Krishna et al. (2022) train an encoder model to score generations given a prefix using contrastive learning. Holtzman et al. (2021) instead propose an alternative PMI-based scoring function to address the "surface form competition" hypothesis, which is related to our inverse-consistency methodology. Prior work in information retrieval (Sachan et al., 2022, 2023) also makes use of a similar idea to re-rank retrieved passages for QA. Our method, however, does not require any training and also demonstrates better accuracy than PMI (see Appendix A.8).

E Appendix: Language Model Prompts

E.1 Question Generation: L2M

Logical program:

```
(AND meteorology.tropical_cyclone (AND (JOIN
meteorology.tropical_cyclone.category (JOIN
meteorology.tropical_cyclone_category.
tropical_cyclones "Tropical Storm Linda")) (JOIN
 meteorology.tropical_cyclone.affected_areas "
turks & caicos islands")))
```

Prompt (for the last L2M iteration):

```
### Instructions:
Translate the following logical form query into
a natural language question in English. The
generated question must have the same meaning as
 the logical query. The generated question must
cover all and only the information present in
the logical query. The generated question should
 use the schema which describes the entities,
relations, and functions present in the logical
query. Use each previous query and solution as a
 hint to solve the next query.

### Logical Query:
(AND meteorology.tropical_cyclone_category (JOIN
 meteorology.tropical_cyclone_category.
tropical_cyclones "Tropical Storm Linda"))
### Schema:
meteorology.tropical_cyclone=tropical cyclone;
meteorology.tropical_cyclone_category=tropical
cyclone category; meteorology.
tropical_cyclone_category.tropical_cyclones=
tropical cyclones
### English Question:
what is the tropical cyclone category of
tropical storm linda?

### Logical Query:
(AND meteorology.tropical_cyclone (JOIN
meteorology.tropical_cyclone.category (JOIN
meteorology.tropical_cyclone_category.
tropical_cyclones "Tropical Storm Linda")))
### Schema:
meteorology.tropical_cyclone=tropical cyclone;
meteorology.tropical_cyclone_category=tropical
cyclone category; meteorology.
tropical_cyclone_category.tropical_cyclones=
tropical cyclones; meteorology.tropical_cyclone.
category=category
### English Question:
what category of tropical cyclone is tropical
storm linda in?

### Logical Query:
(AND meteorology.tropical_cyclone (JOIN
meteorology.tropical_cyclone.affected_areas "
turks & caicos islands"))
### Schema:
meteorology.tropical_cyclone=tropical cyclone;
meteorology.tropical_cyclone.affected_areas=
affected areas
### English Question:
what tropical cyclones have affected the turks &
 caicos islands?

### Logical Query:
(AND (JOIN meteorology.tropical_cyclone.category
 (JOIN meteorology.tropical_cyclone_category.
tropical_cyclones "Tropical Storm Linda")) (JOIN
 meteorology.tropical_cyclone.affected_areas "
turks & caicos islands"))
### Schema:
meteorology.tropical_cyclone=tropical cyclone;
meteorology.tropical_cyclone_category=tropical
cyclone category; meteorology.
tropical_cyclone_category.tropical_cyclones=
tropical cyclones; meteorology.tropical_cyclone.
category=category; meteorology.tropical_cyclone.
affected_areas=affected areas
### English Question:
which tropical cyclones in the tropical storm
linda category have affected the turks & caicos
islands

### Logical Query:
(AND meteorology.tropical_cyclone (AND (JOIN
meteorology.tropical_cyclone.category (JOIN
meteorology.tropical_cyclone_category.
tropical_cyclones "Tropical Storm Linda")) (JOIN
 meteorology.tropical_cyclone.affected_areas "
turks & caicos islands")))
### Schema:
meteorology.tropical_cyclone=tropical cyclone;
meteorology.tropical_cyclone_category=tropical
cyclone category; meteorology.
tropical_cyclone_category.tropical_cyclones=
tropical cyclones; meteorology.tropical_cyclone.
category=category; meteorology.tropical_cyclone.
affected_areas=affected areas
### English Question:
```

Generation (natural language question):

```
all tropical cyclones in the tropical storm
linda categories that have affected the turks &
```

caicos islands

E.2 Question Generation: Inverse-Consistency Re-ranking

Generated question to re-rank:

what is the tropical cyclone category of tropical storm linda?

Prompt:

```
### Instructions:
Translate the following question into its
semantic parse.

### Question:
what is the tropical cyclone category of
tropical storm linda?
### Semantic Parse:
```

Completion to score (logical program):

```
(AND meteorology.tropical_cyclone_category (JOIN
 meteorology.tropical_cyclone_category.
tropical_cyclones "Tropical Storm Linda"))
```

E.3 Reasoning

Test question:

which movies was Rob Williams the writer of?

Prompt:

```
### Instructions:
Write a logical form expression using only
elements mentioned in the provided natural
language question. An "R" before a relation in
the logical expression may be used to indicate a
 reverse or inverse relation.

### Question:
which movies did bernard girard write the script
 for
### Logical Form:
(AND movie.movie (JOIN movie.written_by "Bernard
 Girard"))

### Question:
what movies was paul solet the director of
### Logical Form:
(AND movie.movie (JOIN movie.directed_by "Paul
Solet"))

### Question:
which movies starred amy poehler and were also
written by the people who directed them
### Logical Form:
(AND movie.movie (AND (JOIN movie.starred_actors
 "Amy Poehler") (JOIN movie.written_by (JOIN (R
movie.directed_by) movie.movie))))

### Question:
what movies did matt reeves direct and write the
 script
### Logical Form:
(AND movie.movie (AND (JOIN movie.directed_by "
Matt Reeves") (JOIN movie.written_by "Matt
Reeves")))

### Question:
how many movies did gary k. wolf write the
scripts for
### Logical Form:
(COUNT (AND movie.movie (JOIN movie.written_by "
Gary K. Wolf")))

### Question:
which movies was Rob Williams the writer of
### Logical Form:
```

E.4 Reasoning: Inverse-Consistency Re-ranking

Candidate program to re-rank:

```
(AND travel.travel_destination (JOIN (R book.
book_edition.place_of_publication) (JOIN (R book.
audio_book_reader.audio_books_read) m.09qbn3)))
```

Prompt:

```
### Instructions:
Write a plausible question in English that can
be formed from the provided logical query as a
starting point. The question must contain at
least all of the information present in the
logical query.

### Logical Query:
(AND travel.travel_destination (JOIN (R book.
book_edition.place_of_publication) (JOIN (R book.
audio_book_reader.audio_books_read) m.09qbn3)))
### Plausible Question:
```

Completion to score (test question):

what is the name of the travel destination where mircea cartarescu is published?

F Appendix: Qualitative Examples

F.1 Inverse-Consistency Re-ranking for Question Generation

F.1.1 Re-ranking sequences returned by beam search

```
Program:
(AND religion.founding_figure (JOIN religion.
founding_figure.religion_founded (JOIN religion.
religion.founding_figures "st. peter")))

Standard predictions (top-5, in order of log-
probability scores):
who is paul the apostle?
who founded christianity?
who was the founder of christianity?
who are the founding figures of the religion
founded by st. peter?
who is the founding figure of the religion
founded by st. peter?

Inverse-consistency predictions (top-5, in order
 of inverse-task log-probability scores):
who is the founding figure of the religion
founded by st. peter?
who are the founding figures of the religion
founded by st. peter?
```

```
who was the founder of christianity?
who founded christianity?
who is paul the apostle?
```

F.1.2 Prediction Examples

```
Program:
(COUNT (AND biology.breed_temperament (AND (JOIN
 biology.breed_temperament.breeds (JOIN biology.
animal_breed.place_of_origin ”swiss
confederation”)) (JOIN biology.breed_temperament.
breeds ”Toy Bulldog”))))

Standard prediction:
how many dog breeds are native to switzerland?

Inverse-consistency prediction:
how many are the breed temperaments of the
animal breeds originated from the swiss
confederation and the toy bulldog breed?

---

Program:
(AND medicine.medical_trial (JOIN medicine.
medical_trial.treatment_being_tested ”Stavudine
”))

Standard prediction:
what treatments are being tested in medical
trials?

Inverse-consistency prediction:
in which medical trials is stavudine the
treatment being tested?

---

Program:
(AND medicine.contraindication (JOIN medicine.
contraindication.contraindication_for (JOIN
medicine.medical_treatment.contraindications (
JOIN medicine.contraindication.
contraindication_for ”Teriparatide”))))

Standard prediction:
why is teriparatide contraindicated?

Inverse-consistency prediction:
what are the contraindications for teriparatide?

---

Program:
(AND measurement_unit.volume_unit (JOIN
measurement_unit.volume_unit.measurement_system
(JOIN measurement_unit.measurement_system.
molar_heat_capacity_units ”Joule per mole per
kelvin”)))

Standard prediction:
what is the molar heat capacity of joule per
molecule per kelvin?

Inverse-consistency prediction:
which units of volume have a molar heat capacity
 units of 'joules per mole per kelvin'?
```

F.2 Inverse-Consistency Re-ranking for Reasoning

```
Test Query:
what fictional universe does the harry potter
take place in?

Standard predictions (top-5, in order of log-
probability scores):
(AND fictional_universe.work_of_fiction (JOIN (R
 fictional_universe.fictional_universe.
literary_series_set_here) (JOIN (R
fictional_universe.work_of_fiction.
part_of_these_fictional_universes) m.078ffw)))
(AND fictional_universe.fictional_universe (JOIN
 fictional_universe.fictional_universe.
literary_series_set_here m.078ffw))
(JOIN (R fictional_universe.work_of_fiction.
part_of_these_fictional_universes) m.078ffw)
(AND fictional_universe.fictional_universe (JOIN
 (R book.literary_series.fictional_universe) m
.078ffw))

Inverse-consistency predictions (top-5, in order
 of inverse-task log-probability scores):
(AND fictional_universe.fictional_universe (JOIN
 fictional_universe.fictional_universe.
literary_series_set_here m.078ffw))
(JOIN (R fictional_universe.work_of_fiction.
part_of_these_fictional_universes) m.078ffw)
(AND fictional_universe.fictional_universe (JOIN
 (R book.literary_series.fictional_universe) m
.078ffw))
(AND fictional_universe.work_of_fiction (JOIN (R
 fictional_universe.fictional_universe.
literary_series_set_here) (JOIN (R
fictional_universe.work_of_fiction.
part_of_these_fictional_universes) m.078ffw)))

---

Test Query:
the website which had the api digg api was owned
 by who?

Standard predictions (top-5, in order of log-
probability scores):
(JOIN (R internet.api.site) m.02hz97f)
(JOIN (R internet.website.owner) (JOIN (R
internet.api.site) m.02hz97f))
(JOIN (R internet.api.protocols) m.02hz97f)
(JOIN (R internet.website.owner) (JOIN internet.
website.api (JOIN (R internet.api.protocols) m
.02hz97f)))

Inverse-consistency predictions (top-5, in order
 of inverse-task log-probability scores):
(JOIN (R internet.website.owner) (JOIN (R
internet.api.site) m.02hz97f))
(JOIN (R internet.website.owner) (JOIN internet.
website.api (JOIN (R internet.api.protocols) m
.02hz97f)))
(JOIN (R internet.api.site) m.02hz97f)
(JOIN (R internet.api.protocols) m.02hz97f)

---

Test Query:
name the measurement system that uses newton per
 metre as a surface tension unit.
```

```
Standard predictions (top-5, in order of log-
probability scores):
(JOIN (R measurement_unit.surface_tension_unit.
tension_in_newtons_per_meter) m.02sj4sk)
(JOIN measurement_unit.measurement_system.
surface_tension_units m.02sj4sk)
(AND (JOIN measurement_unit.measurement_system.
surface_tension_units m.02sj4sk) (JOIN (R
measurement_unit.surface_tension_unit.
measurement_system) m.02sj4sk))
(JOIN (R measurement_unit.surface_tension_unit.
measurement_system) m.02sj4sk)


Inverse-consistency predictions (top-5, in order
 of inverse-task log-probability scores):
(JOIN (R measurement_unit.surface_tension_unit.
measurement_system) m.02sj4sk)
(JOIN measurement_unit.measurement_system.
surface_tension_units m.02sj4sk)
(AND (JOIN measurement_unit.measurement_system.
surface_tension_units m.02sj4sk) (JOIN (R
measurement_unit.surface_tension_unit.
measurement_system) m.02sj4sk))
(JOIN (R measurement_unit.surface_tension_unit.
tension_in_newtons_per_meter) m.02sj4sk)

---

Test Query:
kg/m3 is the density units for which system of
measurement?

Standard predictions (top-5, in order of log-
probability scores):
(AND measurement_unit.unit_of_density (JOIN
measurement_unit.unit_of_density.
measurement_system (JOIN measurement_unit.
measurement_system.density_units m.0d1kg)))
(AND measurement_unit.unit_of_surface_density (
JOIN measurement_unit.unit_of_surface_density.
measurement_system (JOIN measurement_unit.
measurement_system.density_units m.0d1kg)))
(JOIN measurement_unit.unit_of_density.
measurement_system (JOIN measurement_unit.
measurement_system.density_units m.0d1kg))
(JOIN measurement_unit.measurement_system.
density_units m.0d1kg)

Inverse-consistency predictions (top-5, in order
 of inverse-task log-probability scores):
(JOIN measurement_unit.measurement_system.
density_units m.0d1kg)
(AND measurement_unit.unit_of_density (JOIN
measurement_unit.unit_of_density.
measurement_system (JOIN measurement_unit.
measurement_system.density_units m.0d1kg)))
(JOIN measurement_unit.unit_of_density.
measurement_system (JOIN measurement_unit.
measurement_system.density_units m.0d1kg))
(AND measurement_unit.unit_of_surface_density (
JOIN measurement_unit.unit_of_surface_density.
measurement_system (JOIN measurement_unit.
measurement_system.density_units m.0d1kg)))

---

Test Query:
what is the name of the exhibition that has the
same exhibition curator with y lle celf?

Standard predictions (top-5, in order of log-
probability scores):
(AND exhibitions.exhibition_curator (JOIN
exhibitions.exhibition_curator.
exhibitions_curated m.0w031yl))
(AND exhibitions.exhibition (JOIN exhibitions.
exhibition.curators (JOIN exhibitions.
exhibition_curator.exhibitions_curated m.0w031yl
)))
(JOIN (R exhibitions.exhibition.curators) m.0
w031yl)
(JOIN exhibitions.exhibition.curators (JOIN
exhibitions.exhibition_curator.
exhibitions_curated m.0w031yl))

Inverse-consistency predictions (top-5, in order
 of inverse-task log-probability scores):
(AND exhibitions.exhibition (JOIN exhibitions.
exhibition.curators (JOIN exhibitions.
exhibition_curator.exhibitions_curated m.0w031yl
)))
(AND exhibitions.exhibition_curator (JOIN
exhibitions.exhibition_curator.
exhibitions_curated m.0w031yl))
(JOIN (R exhibitions.exhibition.curators) m.0
w031yl)
(JOIN exhibitions.exhibition.curators (JOIN
exhibitions.exhibition_curator.
exhibitions_curated m.0w031yl))
```