Don't Generate, Discriminate: A Proposal for Grounding Language Models to Real-World Environments

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

Xiang Deng
The Ohio State University
deng.595@osu.edu

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

Abstract

A key missing capacity of current language models (LMs) is grounding to real-world environments. Most existing work for grounded language understanding uses LMs to directly generate plans that can be executed in the environment to achieve the desired effects. It thereby casts the burden of ensuring grammaticality, faithfulness, and controllability all on the LMs. We propose Pangu, a generic framework for grounded language understanding that capitalizes on the discriminative ability of LMs instead of their generative ability. Pangu consists of a symbolic agent and a neural LM working in a concerted fashion: The agent explores the environment to incrementally construct valid plans, and the LM evaluates the plausibility of the candidate plans to guide the search process. A case study on the challenging problem of knowledge base question answering (KBQA), which features a massive environment, demonstrates the remarkable effectiveness and flexibility of Pangu: A BERT-base LM is sufficient for setting a new record on standard KBQA datasets, and larger LMs further bring substantial gains. Pangu also enables, for the first time, effective few-shot in-context learning for KBQA with large LMs such as Codex.[1]

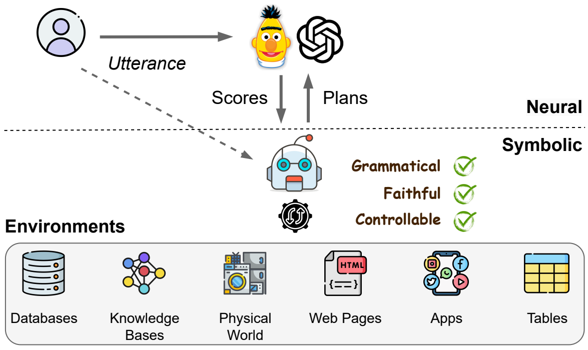

Figure 1: A schematic illustration of the proposed framework, Pangu, where a symbolic agent interacts with the target environment to propose candidate plans, and a neural LM evaluates the plausibility of each plan. The agent searches the environment to incrementally construct the plans, and the LM guides the search process.

1 Introduction

Language models (LMs) such as BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020), and Codex (Chen et al., 2021a) have demonstrated an extraordinary capacity in understanding and generating both natural language (Minaee et al., 2021; Liang et al., 2022) and generic programs (*e.g.*, Python) (Li et al., 2022; Jain et al., 2022; Austin et al., 2021). The recent release of ChatGPT and similar large LMs is elevating this paradigm to a new level. It seems to point us towards a future where natural language serves as a universal device, powered by LMs, for automated problem solving and interacting with the (computing) world.

However, a key missing piece in realizing this future is the connection between LMs and real-world environments, including both digital environments (*e.g.*, databases, knowledge bases, Excel spreadsheets, software, websites, among others) and physical environments (*e.g.*, instruction following robots (Shridhar et al., 2020; Ahn et al., 2022)). Such environments are where many real problems lie. For example, a biologist may need to find all the species of a certain butterfly genus and their geographic distribution from a biology knowledge base, a local grocery store owner may want to visualize the historical sales of different item categories in Excel to decide what and how much to restock before the holiday season, and a physician may need to find patients with specific conditions in a large database of electronic medical records to inform the current diagnosis. *How can LMs enable solving all these problems, which involve seeking information or taking actions in a specific environment, with natural language?*

Each environment is a unique context for interpreting natural language requests from users. *Grounding*, *i.e.*, linking of (natural language) con-

[1] The Pangu library: OSU-NLP-Group/Pangu.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 4928–4949
July 9-14, 2023 ©2023 Association for Computational Linguistics

cepts to contexts (Chandu et al., 2021), therefore becomes the fundamental problem. More precisely, we need to produce a *plan* that can be executed in an environment to achieve the desired effects of the corresponding language request. When a plan is described in a formal language (*e.g.*, SQL for relational databases (Yu et al., 2018) or APIs for web services (Su et al., 2017; Andreas et al., 2020)), it is also called a *program*. The unique challenge of such *grounded language understanding* problems stems from 1) the vast heterogeneity of environments and their planning languages (*e.g.*, SQL, GraphQL/REST APIs, λ-calculus, and robot planning languages), and 2) the vast, oftentimes infinite, number of possible instantiations (or states) of each environment. Some environments can also be dynamic, *e.g.*, a database that is constantly updated or a physical environment with moving objects.

Most existing methods for grounded language understanding follow the popular sequence-to-sequence framework (Sutskever et al., 2014; Cho et al., 2014) and generate the plans/programs in an autoregressive fashion (Xie et al., 2022; Ye et al., 2022; Wang et al., 2021; Song et al., 2022a). A core thesis of this paper is that *directly generating plans may not be the optimal way of using LMs for grounded language understanding*. It requires LMs to have intimate knowledge about each specific planning language and environment, neither of which may be part of an LM's pre-training, to ensure the *grammaticality* and *faithfulness* of the generated plans.[2] The infinite and dynamic environment states also reduce the potential effectiveness of pre-training for improving faithfulness, even if one manages to do so. Furthermore, autoregressive generation with a neural LM lacks fine-grained *control* over planning; it is cumbersome, though not impossible, to factor preferences, business logic, and other values and constraints into the plan generation process. A focus of recent work is to alleviate (some of) these limitations by augmenting autoregressive generation with environment-specific pre-training (Yu et al., 2021; Deng et al., 2021) or constrained decoding (Scholak et al., 2021; Shin et al., 2021; Gu and Su, 2022). However, the fundamental challenges still largely remain.

Mathematically, an LM is simply a joint distribution $p(x_1, x_2, ..., x_n)$ that factors as a product of conditional distributions $\prod_{i=1}^{n} p(x_i|x_1, ..., x_{i-1})$. Existing work leverages the conditional distribution formulation to generate the plan. It thereby casts the burden of ensuring grammaticality, faithfulness, and controllability all on the LM. The main proposal of this paper is to *disentangle LMs from these responsibilities and let LMs be what they originally are—a model that assigns a probability to a sequence of tokens*. In other words, we advocate for using the joint distribution formulation of LMs to evaluate the plausibility of (utterance, candidate plan) pairs instead of directly generating the plan.

To this end, we propose Pangu, a generic framework for grounded language understanding that capitalizes on the discriminative ability of LMs instead of their generative ability (Figure 1).[3] Pangu consists of a symbolic agent and a neural LM working in a concerted fashion. The symbolic agent explores the environment to propose candidate plans, which are guaranteed by design to be both grammatical and faithful. For most real-world environments, due to the size of the search space or partial observability, it is necessary for the agent to *search* in the environment and incrementally extend or refine the plans. The LM plays a key role in this search process—it evaluates the candidate (partial) plans at each search step and guides the agent towards promising search directions; it also determines when the search ends. Finally, it is also easier to control the search process of a symbolic agent than the generation process of a neural LM.

As a case study, we instantiate the proposed framework for complex question answering over knowledge bases (KBQA). KBQA provides an ideal testbed for grounded language understanding because of its massive environment—direct generation with LMs often fails dramatically (Gu et al., 2021). We show that simply using BERT-base with Pangu is sufficient for setting a new record on standard KBQA datasets, and larger LMs further bring substantial gains. Pangu also enables, for the first time, few-shot KBQA by prompting large language models (*e.g.*, Codex): Using only 10 labeled examples, it outperforms all prior methods on GRAPHQ (Su et al., 2016). It provides *unprecedented uniformity* for using LMs—one can easily plug encoder-only LMs, encoder-decoder LMs, or decoder-only LMs into Pangu, through either fine-

[2] We generalize the definition of faithfulness to mean plans that conform to the specifics of an environment such that it can be successfully executed and achieve non-trivial results, *e.g.*, a SQL query that is executable in a specific database and yields a non-empty result set.

[3] Pangu is a primordial being in Chinese mythology who separated heaven and earth. We name our framework after that for its separating the realm of the neural and the symbolic.

tuning or in-context learning. These results highlight the remarkable effectiveness and flexibility of Pangu and validate the proposal of using LMs for discrimination instead of generation.

2 Related Work

2.1 Generation for Grounded Language Understanding

The Seq2Seq framework (Sutskever et al., 2014; Bahdanau et al., 2015) has been the *de facto* choice for grounded language understanding, where the LM directly generates a plan given an input utterance. However, the lack of grounding during pre-training makes generating valid plans from LMs challenging. Recent studies endeavor to alleviate this issue via *input augmentation* or *constrained decoding*. For input augmentation, the environment (or some relevant portion of it) is fed to the LM's encoder together with the utterance (Hwang et al., 2019; Wang et al., 2020; Xie et al., 2022). Such methods rely on the LM to understand the interplay between the language requests and the environment and correctly factor that into plan generation. They therefore require substantial training data to learn and also provide no guarantee for grammaticality or faithfulness. In contrast, constrained decoding methods regulate the decoder's behavior to guarantee grammaticality (Scholak et al., 2021; Shu et al., 2022) or even faithfulness (Liang et al., 2017; Gu and Su, 2022). However, such uses still cast the burden of generating valid plans on the LM itself; controlling the generation process of an LM can be difficult and specific to each planning language and/or environment. In our proposal, the LM is only used to discriminate valid plans proposed by an agent through a controllable search process. More detailed comparison is presented in §5.3.

2.2 Few-Shot Grounded Language Understanding with LLMs

Large language models (LLMs) (Brown et al., 2020; Chen et al., 2021a) have demonstrated strong few-shot learning capabilities in various tasks, from writing programs to query structured and unstructured data (Austin et al., 2021; Rajkumar et al., 2022; Cheng et al., 2022), interacting with online websites (Gur et al., 2022; Nakano et al., 2021), to generating procedural plans and guiding embodied agents in virtual environments (Singh et al., 2022; Ahn et al., 2022; Shah et al., 2022; Song et al., 2022b). Most existing work still capitalizes on the generative ability of LLMs. A common strategy to encourage an LLM to produce valid plans is to directly *describe* the environment in the LLM's context (*i.e.*, input augmentation), which is difficult for complex environments like KBs. A concurrent work of ours (Li et al., 2023b) asks the LLM to directly generate a proxy plan from the input question without the environment description, which is then used to retrieve a valid plan from a set of candidate plans. However, this design is tailored specifically to the KB query language and is limited to generating plans with at most two hops due to the combinatorial explosion in their candidate enumeration. In contrast, Pangu shields the LLM from the complexity of the environment and lets the LLM focus on evaluating the plausibility of candidate plans proposed by an agent. One interesting related work is Ahn et al. (2022), where an LLM is used to score atomic action (skill) proposals, which are guaranteed to conform to affordance constraints, from an embodied agent. Pangu shares a similar spirit of using LMs for discrimination, but we support more complex plans through a search process in the environment guided by an LM.

2.3 Bottom-Up Semantic Parsing

Our instantiation of Pangu on KBQA is closely connected to bottom-up semantic parsing, particularly SmBoP (Rubin and Berant, 2021), a text-to-SQL model that iteratively constructs a complex plan from a set of subplans. Pangu similarly constructs a complex plan incrementally from smaller subplans, but it makes the following main departures. First, SmBoP requires all ingredients (*i.e.*, column headers, table names, and DB values) at the beginning of parsing. This assumption does not generally hold for more complex or partially observable environments, where ingredients need to be discovered through search. In our method, only topic entities are needed as the initial plan, which can be readily obtained using an entity linker (Li et al., 2020). Second, our scoring function is based on a straightforward application of LMs, while SmBoP uses a more intricate architecture with extra parameters. Also related is an array of earlier KBQA methods that adopt an enumerate-and-rank approach (Yih et al., 2015; Gu et al., 2021; Ye et al., 2022). Because they try to enumerate all candidate plans up front, the maximum plan complexity is bound to be small. Our adaptive search process allows for flexible construction of more complex plans.

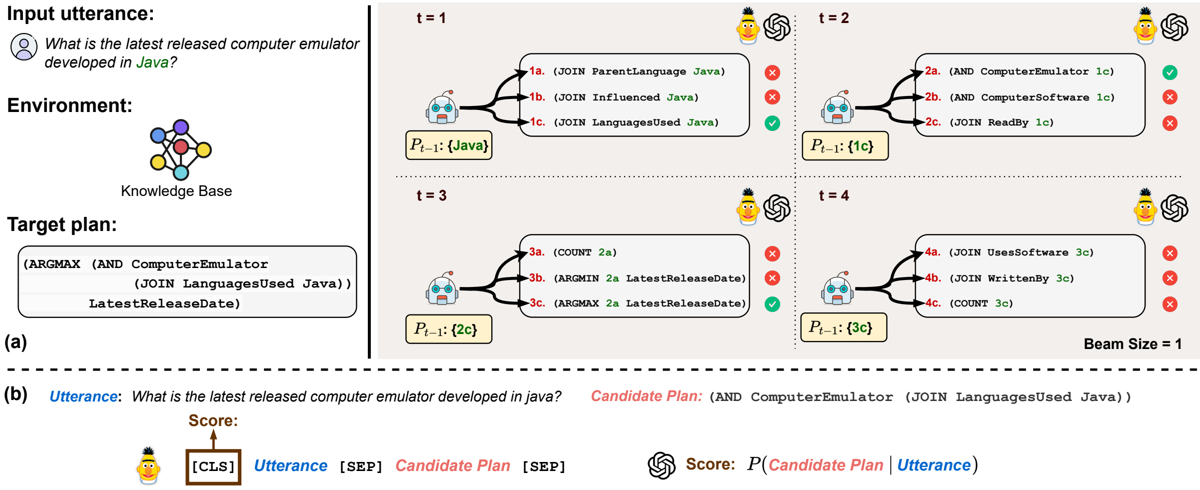

Figure 2: (a) An illustration of how an agent collaborates with an LM to incrementally produce a complex target plan over a KB using beam search (beam size $= 1$ in this example). At each step, the agent extends the current plans based on the environment to produce new candidate plans. An LM then scores the candidate plans and returns the top-ranked ones. The search process terminates when there is no candidate plan that scores higher than the current best plan (*e.g.*, $4a$-c are all worse than $3c$). (b) Using different LMs (*left*: BERT, *right*: Codex) to evaluate the plausibility of plan $2a$. It resembles using LMs for semantic matching between the utterance and the plan.

3 Approach

An overview of the Pangu framework is presented in Algorithm 1. An overarching assumption of Pangu is that a complex plan can be incrementally constructed by an agent through its exploration in an environment. Such an agent can be a robot doing household tasks in a physical environment (Shridhar et al., 2020), or a virtual agent that orchestrates API calls of different web services (Andreas et al., 2020) or traverses a database/KB (Yu et al., 2018; Gu et al., 2022). Starting from a set of initial plans P_0 (may be empty), at each step, the agent interacts with the environment E to extend the current plans into a new set of candidate plans (line 4). The candidate plans are guaranteed to be *valid* (*i.e.*, both grammatical and faithful). An LM then scores the candidate plans, and the top K (the beam size) plans are retained for further exploration in the next step (line 5). The same procedure loops until a termination check is passed (line 6); the best plan is then returned.

Pangu mainly shines in that a symbolic agent explores the environment to propose valid plans and shields the LM from having to handle the large search space for valid plan generation. Instead, the LM only focuses on evaluating the plausibility of the proposed plans. An LM can be easily fine-tuned to excel at this assignment, or, in the case of LLMs such as Codex, they come with such ability out of the box, which enables few-shot in-context learning. Pangu is a generic framework and can potentially accommodate many grounded language understanding tasks by instantiating the various functions in Algorithm 1 accordingly. Next, we discuss our instantiation on KBQA. More discussion on Pangu's applicability to other tasks, with preliminary results, can be found in Appendix A.

Algorithm 1: PANGU

1 **Input:** utterance q, initial plans P_0, environment E
2 $t \leftarrow 1$;
3 **while** *True* **do**
  // Agent proposes plans
4 $C_t \leftarrow$ **Candidate-Plans**(P_{t-1}, E)
  // LM scores and prunes plans
5 $P_t \leftarrow$ **Top-**$K(q, C_t)$
6 **if** ***Check-Termination***() = *True* **then**
7   **return** top-scored plan
8 $t \leftarrow t + 1$

3.1 KBQA: Preliminaries

Without loss of generality, we use KBs as our target environment and the KBQA task as a concrete example for ease of discussion. It is an ideal testbed because of the massive environment provided by modern KBs (*e.g.*, FREEBASE (Bollacker et al., 2008) contains 45 million entities and 3 billion facts for over 100 domains), which makes grounding particularly challenging. Given a KB $\mathcal{K} \subset \mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L} \cup \mathcal{C})$, where $\mathcal{C}$ is a set of classes, $\mathcal{E}$ a set of entities, $\mathcal{L}$ a set of literals and

$\mathcal{R}$ a set of binary relations, the task of KBQA is to find a set of answer entities to an input utterance in the KB. KBQA is typically modeled as semantic parsing (Gu et al., 2022), where the utterance is mapped to an executable program/plan in a certain formal language (e.g., SPARQL, λ-calculus, or S-expression) whose denotation is the answer. We use S-expressions (Gu et al., 2021) for its compactness. An example is shown in Figure 2.

3.2 Candidate Plan Enumeration

To handle the large search space, the agent casts the task as a step-wise decision-making problem. A plan for KBQA can be decomposed into a nested sequence of subplans (Gu and Su, 2022) (Figure 2). The *length* of a plan is defined as the number of atomic subplans it contains.

For KBQA, P_0 can be a set of entity proposals (*e.g.*, {Java}) obtained using off-the-shelf entity linkers (Li et al., 2020). At step t, the agent considers P_{t-1}, the length $t-1$ plans, and decides how to further extend them into C_t, the valid plans of length t, based on the environment. This often involves executing the current plans in the environment. Consider the example in Figure 2 at $t=1$, the agent finds all the relations connected to Java and enumerates all the length-1 valid plans. The LM scores the candidate plans and prunes all but the top-ranked plan because beam size is 1. At $t=2$, the agent executes plan $1c$ to get its denotation (*i.e.*, a set of entities) in the KB, based on which the agent further discovers the relations and classes (*e.g.*, ComputerEmulator, ComputerSoftware, and ReadBy) connected to those entities to form valid length-2 plans. All the plans produced in this process are guaranteed to be valid. See Appendix B for a more detailed discussion of this process.

3.3 LM-Based Scoring

After the agent enumerates a set of candidate plans, an LM assists with its decision making by evaluating the plausibility of each candidate plan. The interface for evaluating a plan using LMs resembles using LMs for semantic matching: Given a pair of (u: *utterance*, $c \in C_t$: *candidate plan*), an LM acts as a scoring function: $s(u,c) \rightarrow \mathbb{R}$, which indicates to what extent the candidate plan matches the intent of the utterance. The plausibility of a candidate oftentimes can be indicated by simple linguistic cues, *e.g.*, ComputerEmulator in $2a$ might be a strong indicator (Figure 2(a)).

We follow the common practice of using LMs for semantic matching. For encoder-only LMs like BERT, we directly get a score from the representation of the [CLS] token (Figure 2(b)). For encoder-decoder LMs like T5 (Raffel et al., 2020), we follow Zhuang et al. (2022) to feed both the utterance and the candidate plan to the encoder and let the decoder decode only for one step. The decoding probability over an token that is unused during pre-training is then repurposed as a proxy for matching score.[4] For decoder-only LMs like Codex, we model the score as the probability of generating the candidate plan conditioned on the utterance, *i.e.*, $P(c|u)$. Intuitively, a good scoring function s should respect the following *partial order*:

$$\begin{aligned} &s(u,c_1) > s(u,c_2), \quad \forall c_1 \in G_t \text{ and } \forall c_2 \in G_{t-1}, \\ &s(u,c_1) > s(u,c_2), \quad \forall c_1 \in G_t \text{ and } \forall c_2 \in C_t \backslash G_t, \\ &s(u,c') > s(u,c_i), \quad \forall c_i \neq c' \end{aligned}$$

where G_t is the set of gold (sub-)plans at step t (*i.e.*, length-t subplans of the target plan), $C_t \backslash G_t$ is the set of length-t candidate plans except the gold (sub-)plans, and c' is the target plan.

In other words, a gold subplan should be scored higher than (1) any negative (*i.e.*, not gold) plans at the same step (*e.g.*, $2a$ should be scored higher than $2c$), because they contain information irrelevant to u, and (2) any gold sub-plans of length $< t$ (*e.g.*, $2a$ should be scored higher than $1c$) because they are less complete. In addition, c' should be scored higher than any other plan.

3.4 Termination Check

Assuming the LM can assign reasonable scores to candidate plans following the above partial order, we can naturally define the condition for termination in Algorithm 1: It terminates if the highest score of candidate plans at step t is lower than the highest score of candidate plans at step $t-1$, which, ideally, should indicate no reachable candidate plan of length $\geq t$ is better than the plans at step $t-1$, and thus the search process terminates.

3.5 Learning

We discuss the learning procedure for both fine-tuning LMs (*e.g.*, BERT and T5) and in-context learning with LLMs (*e.g.*, Codex). For both settings, we use pairs of utterances and gold plans for supervision.

[4] We use `<extra_id_23>` as the proxy token for T5.

Fine-tuning. Given a gold plan of length T, we first derive its gold sub-plans G_t of each step $t \le T$ (*e.g.*, $1c$ for step 1 and $2a$ for step 2 in Figure 2). Fine-tuning proceeds with beam search similar to the test-time behavior, but with bottom-up teacher forcing (Williams and Zipser, 1989; Rubin and Berant, 2021), *i.e.*, the gold plans of the current step should always be inserted into the beam. At each step of beam search, we get the probability of each candidate plan $c \in C_t$ with softmax over the scores: $p(c) = \text{softmax}\{s(u, c)\}_{c \in C_t \cup G_{t-1}}$. G_{t-1} is also included here to encourage LMs to explicitly learn the partial order by minimizing the loss:

$$-\frac{1}{Z}\sum_{t=1}^{T+1}\sum_{c \in C_t} \hat{p}(c) \log p(c)$$

where Z is the total number of summed items, and $\hat{p}(c)$ equals 1 if $c \in G_t$ and 0 elsewise. Note that, for the $T+1$ step, we let $G_{T+1} = G_T$. This additional step aims to enforce the third condition in the partial order. Our objective is essentially a listwise learning-to-rank objective based on the cross entropy (Cao et al., 2007).

In-Context Learning. We directly use pairs of utterances and gold plans as in-context demonstrations to the LLM, with a simple task instruction in the prompt: *"Please translate the following questions to Lisp-like programs."* The LLM is therefore expected to capture the desired partial order by observing the in-context examples. For concrete examples of prompts, please refer to Appendix F. When scoring using LLMs, we normalize the likelihood *w.r.t.* the number of tokens in the plan to handle plans of varying lengths.

4 Experimental Setup

4.1 Datasets

We experiment with three KBQA datasets of different scale and nature (statistics in Table C.3).

GRAILQA (Gu et al., 2021) is a large-scale dataset that evaluates three levels of generalization, namely, *i.i.d.*, *compositional* (novel compositions of seen constructs), and *zero-shot* (totally novel domains). It also features diverse questions of different complexity (*e.g.*, programs may involve up to 4 relations) and aggregation functions (*e.g.*, comparatives, superlatives, and counting).

GRAPHQ (Su et al., 2016) is a moderate-scale dataset. Due to the small size of its training set and the non-i.i.d. setting, GRAPHQ is particularly challenging. In our experiments, we use the processed version by Gu and Su (2022), which maps the original dataset from FREEBASE 2013-07 to FREEBASE 2015-08-09.

WEBQSP (Yih et al., 2016) is a moderate-scale dataset with questions from Google query logs. It mainly tests i.i.d. generalization on simple questions. It is a clean subset of WEBQ (Berant et al., 2013) with program annotations.

The gold programs for all three datasets are provided in S-expressions (Gu and Su, 2022), which can be determinstically converted into SPARQL queries to get final execution results.

4.2 Baselines

We mainly compare Pangu with state-of-the-art baselines that use LMs as a generative model, including ArcaneQA (Gu and Su, 2022), TIARA (Shu et al., 2022), DecAF (Yu et al., 2022), and RnG-KBQA (Ye et al., 2022). Constrained decoding (*i.e.*, ArcaneQA and TIARA) and input augmentation (*i.e.*, TIARA, DecAF) are used to enhance plan generation. Also, the last three models use a combination of LMs for multiple purposes (*i.e.*, retrieval/ranking/decoding). In addition, we also compare with UnifiedSKG (Xie et al., 2022). UnifiedSKG assumes a set of schema items are provided as input, where the gold schema items are always included and the number of negative schema items is restricted to 20 for GRAILQA. It is thus a less fair comparison for other methods, but we include it anyway because it is a representative way of autoregressive plan generation using an LLM. Compared with the baselines, Pangu requires no extra parameter, no modification to the LM, and no need to combine multiple LMs. Pangu provides unprecedented uniformity of using LMs of different nature. More details on baselines can be found in Appendix C.2.

4.3 Implementation Details

For the fine-tuning experiments, we experiment with BERT-base, T5-base, T5-large, and T5-3B, and use the full training set of each dataset for fine-tuning. For the in-context learning experiments, we experiment with Codex.[5] We randomly sample 10/100/1,000 training examples from each dataset and use that as the pool for dynamic retrieval. During inference, for each test example, we retrieve

[5]We opt for Codex because it is free, but small-scale experiments also show competitive performance from ChatGPT.

Model	Overall EM	Overall F1	I.I.D. EM	I.I.D. F1	Compositional EM	Compositional F1	Zero-shot EM	Zero-shot F1	Dev Overall EM	Dev Overall F1
QGG (Lan and Jiang, 2020)	–	36.7	–	40.5	–	33.0	–	36.6	–	–
BERT+Ranking (Gu et al., 2021)	50.6	58.0	59.9	67.0	45.5	53.9	48.6	55.7	–	–
ReTraCk (Chen et al., 2021b)	58.1	65.3	84.4	87.5	61.5	70.9	44.6	52.5	–	–
RnG-KBQA (Ye et al., 2022)	68.8	74.4	86.2	89.0	63.8	71.2	63.0	69.2	71.4	76.8
ArcaneQA (Gu and Su, 2022)	63.8	73.7	85.6	88.9	65.8	75.3	52.9	66.0	69.5	76.9
Uni-Parser (Liu et al., 2022)	69.5	74.6	85.5	88.5	65.1	71.1	64.0	69.8	70.8	76.5
TIARA (Shu et al., 2022)	73.0	78.5	**87.8**	**90.6**	69.2	76.5	68.0	73.9	75.3	81.9
DecAF (Yu et al., 2022)	68.4	78.7	84.8	89.9	73.4	81.8	58.6	72.3	–	81.4
UnifiedSKG w/ T5-3B (Xie et al., 2022)	–	–	–	–	–	–	–	–	70.1*	–
Pangu (this work)										
w/ BERT-base	73.7	79.9	82.6	87.1	74.9	81.2	69.1	76.1	75.0	82.1
w/ T5-base	73.6	79.9	84.7	88.8	73.1	80.1	68.6	75.8	**76.0**	82.8
w/ T5-large	74.8	81.4	82.5	87.3	**75.2**	**82.2**	71.0	78.4	75.8	83.3
w/ T5-3B	**75.4**	**81.7**	84.4	88.8	74.6	81.5	**71.6**	**78.5**	75.8	**83.4**
w/ Codex (10-shot)	48.9	56.3	51.8	58.1	43.3	51.2	50.1	57.8	–	–
w/ Codex (100-shot)	53.3	62.7	54.7	62.9	54.5	63.7	52.3	62.2	–	–
w/ Codex (1000-shot)	56.4	65.0	67.5	73.7	58.2	64.9	50.7	61.1	–	–

(a) GRAILQA

Model	F1
UDEPLAMBDA (Reddy et al., 2017)	17.7♯
PARA4QA (Dong et al., 2017)	20.4♯
SPARQA (Sun et al., 2020)	21.5♯
BERT+Ranking (Gu et al., 2021)	27.0
ArcaneQA (Gu and Su, 2022)	34.3
Pangu (this work)	
w/ BERT-base	52.0
w/ T5-base	53.3
w/ T5-large	55.6
w/ T5-3B	**62.2**
w/ Codex (10-shot)	42.8
w/ Codex (100-shot)	43.3
w/ Codex (1000-shot)	44.3

(b) GRAPHQ

Model	F1
QGG (Lan and Jiang, 2020)	74.0
ReTraCk (Chen et al., 2021b)	71.0
CBR (Das et al., 2021)	72.8
Program Transfer (Cao et al., 2022)	76.5*
RnG-KBQA (Ye et al., 2022)	75.6
ArcaneQA (Gu and Su, 2022)	75.6
Uni-Parser (Liu et al., 2022)	75.8
TIARA (Shu et al., 2022)	76.7
DecAF (Yu et al., 2022)	78.8
Pangu (this work)	
w/ BERT-base	77.9
w/ T5-base	77.3
w/ T5-large	78.9
w/ T5-3B	**79.6**
w/ Codex (10-shot)	45.9
w/ Codex (100-shot)	54.5
w/ Codex (1000-shot)	68.3

(c) WEBQSP

Table 1: Overall results. Pangu achieves a new state of the art on all three datasets and shows great flexibility in accommodating LMs of different nature. Also, for the first time, Pangu enables effective few-shot in-context learning for KBQA with Codex. * using oracle entity linking. ♯ results on the original GRAPHQ 2013-07, otherwise it uses the version from Gu and Su (2022), which is a slightly smaller subset. All baselines after 2020 are trained using gold programs in S-expressions.

10 in-context examples from the pool using BM25-based utterance similarity. We use entity linking results from off-the-shelf entity linkers. More details on implementations can be found in Appendix C.3.

5 Results

5.1 Main Results

Fine-tuning results. The main results are shown in Table 1. Using a BERT-base LM, Pangu already achieves a new state of the art on GRAILQA and GRAPHQ, and only trails behind DecAF on WEBQSP, which uses a 3B-parameter LM. On GRAPHQ, Pangu with BERT-base dramatically improves the state-of-the-art F1 from 31.8% to 48.2%. These are strong evidence for Pangu being a better protocol for using LMs for grounded language understanding. Pangu's strong generalizability with limited training data is also confirmed by its performance on the zero-shot generalization of GRAILQA. Our method also shows an unprecedented uniformity in accommodating different LMs (encoder-only, encoder-decoder, decoder-only, through both fine-tuning and in-context learning) and a reliable return from model size—using increasingly larger LMs yields monotonically improved results across the board, with T5-3B setting the new state of the art on all datasets. One interesting observation is that Pangu slightly underperforms on the i.i.d. subset of GRAILQA. It turns out that, because the discriminative task is much

Question I	*"neil leslie diamond composed what tv song?"*
Pangu	(AND tv.tv_song (JOIN music.composition.composer m.015_30)) (✓)
ArcaneQA	(AND music.recording (**JOIN** music.recording.song (JOIN music.composition.composer m.015_30))) (✗)
ArcaneQA△	(JOIN music.composition.composer m.015_30) (**JOIN** music.recording.song #0) (AND music.recording #1)
Question II	*"which software falls into both continuous integration and build automation genres?"*
Pangu	(AND computer.software (AND (JOIN computer.software.software_genre m.05vvqy) (JOIN computer.software.software_genre m.0h2vrf))) (✓)
ArcaneQA	(**AND** computer.software (JOIN computer.software.software_genre m.05vvqy)) (✗)
ArcaneQA△	(JOIN computer.software.software_genre m.05vvqy) (**AND** computer.software #0)

Table 2: Two representative examples that Pangu succeeds while ArcaneQA fails, both w/ BERT-base. △ denotes the original order of the decoder's output. The first incorrect token predicted by ArcaneQA is marked in **red**.

easier for LMs to learn than the generative task, Pangu converges very fast (at most two epochs) and gets fewer training steps for overfitting the i.i.d. setting, in exchange for better non-i.i.d. generalization. The strong performance on WEBQSP, an i.i.d. dataset, further supports this observation, because now Pangu can more sufficiently fit the i.i.d. training data.

In-context learning results. For the first time, we show the feasibility of effective few-shot KBQA with LLMs. On GRAILQA, Pangu with Codex achieves an overall F1 of 56.3% with only 10 training examples. Though there is still a gap to the full-data fine-tuning results, it is still impressive, especially considering the massive meaning space of the KB. On GRAPHQ, Pangu with Codex even outperforms ArcaneQA using 10 training examples. This further confirms that Pangu is particularly strong in generalizing to new environments with limited training data. On WEBQSP, Pangu trails behind fine-tuning methods when only using 10 training examples; however, increasing the size of the pool for retrieval can significantly boost the performance, which is expected given WEBQSP's i.i.d. nature. While for non-i.i.d. datasets like GRAILQA and GRAPHQ, the gain from more training examples is marginal.

Fine-grained performance decomposition by question complexity can be found in Appendix D, which show that Pangu works well across questions of different complexity.

5.2 Sample Efficiency Analysis

Intuitively, by using LMs for discrimination instead of generation, the task becomes easier for LMs and thus improves their sample efficiency. Our sample efficiency experiments in Figure 3 confirm this hypothesis. We downsample GRAILQA's training data and randomly sample 1, 10, 100, and 1,000 training examples and report the results on 500 random dev examples. We compare Pangu with ArcaneQA and UnifiedSKG using the same LMs. We use oracle entity linking to have a more direct comparison with UnifiedSKG (though UnifiedSKG still has an unfair advantage as previously mentioned). In addition, we also include Pangu with Codex and use the downsampled training set as the pool for retrieval. First, we observe that, when both using T5-base, UnifiedSKG significantly underperforms Pangu. The main reason is that most predicted plans by UnifiedSKG are invalid in the low-data regime. ArcaneQA uses constrained decoding to alleviate this issue, but still consistently underperforms Pangu when both using BERT-base. For in-context learning using Codex, Pangu achieves an EM of over 50% *with only one training instance*. It consistently outperforms all fine-tuning models under low-data settings (*i.e.*, less than 1,000 training examples). Compared with UnifiedSKG, Pangu shows both stronger performance and better robustness against different training data selections.

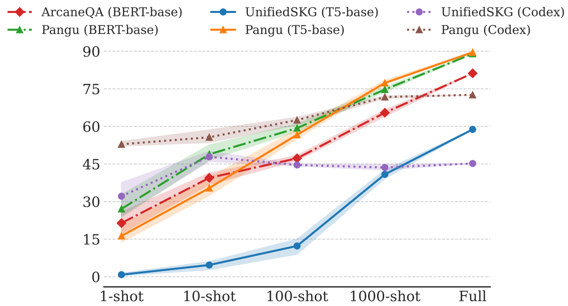

Figure 3: Sample efficiency results. We conduct three runs with different training examples and show the mean EM; shaded areas denote max/min.

5.3 Pangu vs. Constrained Decoding

To better understand Pangu's advantage over generation-based methods, we compare Pangu with

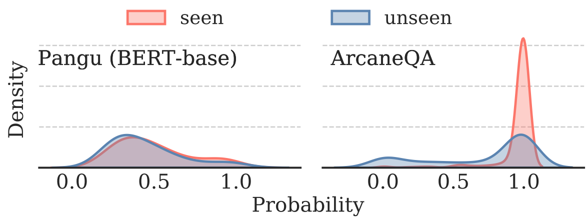

Figure 4: Distribution of the probabilities assigned to predicted programs that are seen and unseen during training. We use kernal density smoothing for better visualization, so the x-axis goes over 1.0.

ArcaneQA. ArcaneQA is the only open-source baseline that uses constrained decoding to enforce the validity of predicted plans. There are two main reasons for Pangu's superiority. First, though constrained decoding can also help ensure plan validity, the autoregressive decoder operates with token-level local normalization and thus lacks a global view. As a result, local failures may break its predictions. For example, a wrong local prediction (e.g., function name) by ArcaneQA leads to catastrophic errors (Table 2). By evaluating candidate *plans* instead of candidate *tokens*, Pangu has a more global view and is less likely to make such local errors. Second, Pangu is less susceptible to overfitting and thus achieves better performance in non-i.i.d. settings. Pangu does not learn to generate a plan; instead, it learns to evaluate the plausibility of utterance-plan pairs. Such knowledge is more transferable. An interesting observation is shown in Figure 4, where Pangu's output probability distributions are consistent across programs seen and unseen in training. For ArcaneQA, however, there is a drastic shift from seen to unseen. This is also consistent with prior findings that autoregressive models tend to overfit seen structures during training by Bogin et al. (2022). It makes non-i.i.d. generalization more difficult.

We also conduct an error analysis in Appendix E, which sheds some light on future improvements.

6 Conclusions

In this paper, we proposed to capitalize on the discriminative ability of language models (LMs) instead of their generative ability for grounded language understanding. Building on this proposal, we proposed a generic framework, Pangu, which consists of a symbolic agent and a neural LM working in a concerted fashion and creates a better separation between the realm of the neural and the symbolic. This work opens the door for developing versatile and sample-efficient grounded language understanding systems that fully capitalize on the language understanding ability of LMs while avoiding their limitations. It also sheds light on developing better neuro-symbolic systems in general.

Limitations

Despite the strong performance of Pangu, we identify several limitations that call for further improvement. The first major limitation lies in efficiency. Because Pangu requires an LM to iteratively score candidate plans, it is resource-consuming in terms of both time and computing. Compared with ArcaneQA, which efficiently handles complex questions in KBQA, Pangu is about twice as slow for both training and inference and consumes about twice as much GPU memory when using the same LM. Concretely, to predict a plan of L tokens, generation-based methods involve using an LM to do L forward passes. For Pangu, the number of forward passes is proportional to the number of candidate plans, which can range widely. In the future, algorithms with complexity better than $O(N)$, N being the number of candidate plans, are desired to find the top-K candidates. That being said, we would like to note that both ArcaneQA and Pangu are more efficient than most existing methods due to their efficient dynamic search design. For example, Pangu is 8 times faster than RnG-KBQA, according to the numbers reported in Gu and Su (2022). Nonetheless, we list efficiency as a limitation because there is clear potential for further improvement.

Second, though Pangu has shown some promising results with Codex, the true potential of enabling few-shot grounded language understanding with Pangu has yet to be fully realized. We only experiment with a straightforward scoring function and have not experimented with different prompt designs systematically. In the future, we plan to try different prompt designs, retrievers, and scoring functions, including using latest techniques like chain-of-thought prompting (Wei et al., 2022).

Third, though orthogonal to the general framework of our proposal, in our current instantiation, we assume gold plans for training. However, gold plans can be expensive to collect for some environments. Exploring fine-tuning LMs with weak supervision can be an interesting direction. In addition to proposing candidate plans to the LM, the agent may also respond to the LM with rewards based on its decisions (Liang et al., 2017).

Finally, one important merit of Pangu, controllability, is under-explored in this paper, because it is not very necessary for KBQA. While for tasks like text-to-SQL parsing, controllability could be a highly desirable property. Intruders may manipulate text-to-SQL models to launch database attacks via SQL injection (Peng et al., 2022). With Pangu, we can easily get rid of malicious SQL operations in candidate enumeration. However, for generation-based methods, such controls are hard to achieve during generation because the decoding process can be shortsighted—it is difficult to tell whether the current predicted token would lead to a malicious operation several steps later. We leave exploration on Pangu's controllability to future work.

Ethics Statement

LLMs are no longer just a laboratory curiosity; they are being used in real-world systems to interact with real-world environments (both digital and physical). To ensure successful deployment of LLMs in these scenarios, it is essential to improve their controllability, as failure to do so could lead to catastrophic results. In digital environments, such as databases, unexpected behavior could lead to safety issues with a company's data and property. In physical environments, it could even put human life at risk. Pangu is proposed to provide better controllability for LLMs when being depolyed to interact with different environments. Specifically, safety considerations can be explicitly incorporated into the agent's candidate proposal (the symbolic part of Pangu) for enhanced security (*i.e.*, harmful actions are directly excluded from the candidates pool).

Acknowledgements

The authors would like to thank Percy Liang, Jiawei Han, Jonathan Berant, Huan Sun, and other colleagues from the OSU NLP group for their valuable feedback. The authors would also like to thank Shijie Chen and Chan Hee Song for proof-of-concept implementation of Pangu on other tasks, Yiheng Shu for sharing their entity linking results, and Tianbao Xie for clarifications on UnifiedSKG. This research was supported in part by ARL W911NF2220144, NSF OAC 2112606, and Ohio Supercomputer Center (Center, 1987).

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *CoRR*, abs/2204.01691.

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

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program synthesis with large language models. *CoRR*, abs/2108.07732.

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

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

Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. 2022. A persistent spatial semantic representation for high-level natural language instruction execution. In *Conference on Robot Learning*, pages 706–717. PMLR.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. *CoRR*, abs/2201.05899.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8128–8140, Dublin, Ireland. Association for Computational Linguistics.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 129–136. ACM.

Ohio Supercomputer Center. 1987. Ohio supercomputer center.

Khyathi Raghavi Chandu, Yonatan Bisk, and Alan W Black. 2021. Grounding 'grounding' in NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4283–4305, Online. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

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

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Binding language models in symbolic languages. *CoRR*, abs/2210.02875.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

E. F. Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387.

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

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

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

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: three levels of generalization for question answering on knowledge bases. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3477–3488. ACM / IW3C2.

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. Knowledge base question answering: A semantic parsing perspective. In *4th Conference on Automated Knowledge Base Construction*.

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.

Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. 2022. Understanding HTML with large language models. *CoRR*, abs/2210.03945.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *CoRR*, abs/1902.01069.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Naman Jain, Skanda Vaidyanath, Arun Shankar Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram K. Rajamani, and Rahul Sharma. 2022. Jigsaw: Large language models meet program synthesis. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pages 1219–1231. ACM.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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

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.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Decoupling the skeleton parsing and schema linking for text-to-sql. *CoRR*, abs/2302.05965.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A simple and performant baseline for vision and language. *CoRR*, abs/1908.03557.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023b. Few-shot in-context learning for knowledge base question answering. *CoRR*, abs/2305.01750.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

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

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. Holistic evaluation of language models. *CoRR*, abs/2211.09110.

Ye Liu, Semih Yavuz, Rui Meng, Dragomir Radev, Caiming Xiong, and Yingbo Zhou. 2022. Uni-Parser: Unified semantic parser for question answering on knowledge base and database. *CoRR*, abs/2211.05165.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13–23.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3).

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. WebGPT: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

Alexander Pashevich, Cordelia Schmid, and Chen Sun. 2021. Episodic transformer for vision-and-language navigation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 15922–15932. IEEE.

Xutan Peng, Yipeng Zhang, Jingfeng Yang, and Mark Stevenson. 2022. On the security vulnerabilities of text-to-sql models. *CoRR*, abs/2211.15363.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *CoRR*, abs/2204.00498.

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

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

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dhruv Shah, Błażej Osiński, brian ichter, and Sergey Levine. 2022. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *6th Annual Conference on Robot Learning*.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. IEEE.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. 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.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2022. Progprompt: Generating situated robot task plans using large language models. *CoRR*, abs/2209.11302.

Chan Hee Song, Jihyung Kil, Tai-Yu Pan, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022a. One step at a time: Long-horizon vision-and-language navigation with milestones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15482–15491.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022b. LLM-Planner: Few-shot grounded planning for embodied agents with large language models. *CoRR*, abs/2212.04088.

Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark J. Encarnación. 2017. Building natural language interfaces to web apis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, *CIKM 2017, Singapore, November 06 - 10, 2017*, pages 177–186. ACM.

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.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. SPARQA: skeleton-based semantic parsing for complex questions over knowledge bases. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8952–8959. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

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

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir R. Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *CoRR*, abs/2201.05966.

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.

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

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. *CoRR*, abs/2210.00063.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2021. GraPPa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. RankT5: Fine-tuning T5 for text ranking with ranking losses. *CoRR*, abs/2210.10634.

Appendices

In this supplementary material, we provide further details as follows:

- Appendix A: Broader Applicability of Pangu
- Appendix B: Candidate Enumeration
- Appendix C: Experimental Setup
- Appendix D: Decomposition by Question Complexity
- Appendix E: Error Analysis
- Appendix F: Examples of Prompts

A Broader Applicability of Pangu

Algorithm 1 describes a generic framework for grounded language understanding, but the concrete implementation for the functions in Algorithm 1 may vary for different tasks. We have shown a representative instantiation for KBQA. In this section, we briefly discuss the possible instantiation for two other tasks of different nature. In addition, we also present some preliminary results we have obtained to demonstrate the feasibility of Pangu.

A.1 Text-to-SQL Parsing

Possible Instantiation. Similar to KBQA, Text-to-SQL parsing also aims to map a natural language utterance into a program that can be executed over a relational database (instead of a KB). When the database schema is reasonably small (generally true for existing datasets like Spider (Yu et al., 2018)), we can define P_0 as the set of all schema items (*i.e.*, column headers and table names) plus the set of cell values mentioned in the utterance, which should be straightforward to identify (*e.g.*, with string matching). In this way, the agent can construct candidate programs similarly to Rubin and Berant (2021).[6] The termination check for text-to-SQL parsing can also be implemented similarly.

Preliminary Results. We test Pangu on Spider, a popular benchmark on Text-to-SQL parsing, following the aforementioned instantiation. Due to the smaller environment of relational databases (compared with KBs), schema linking performance on Spider is already at 99% (Li et al., 2023a). We therefore start the search assuming ingredients have been identified (i.e., the initial plan). With the same language model being used (*i.e.*, CodeBERT (Feng et al., 2020)), we achieved 70.6% Exact-Match on Spider dev, comparable to SmBop's 71.7% (Rubin and Berant, 2021), a strong bottom-up parser baseline, when using the same LM. Note that these are only preliminary results to demonstrate the feasibility of applying Pangu to other tasks. There is still a large room to improve by, e.g., optimizing the search process or using stronger LMs.

A.2 Interacting with Real-World Environments

Possible Instantiation. Pangu can also be used for guiding bots that interact with real-world environments, both online websites (Gur et al., 2022; Nakano et al., 2021) and physical environments through embodied agents (Shridhar et al., 2020). Given a complex task to be accomplished in the environment, an agent may decompose it into a sequence of subplans (*e.g.*, making a cup of coffee entails first finding a cup then picking up the cup, etc.; Song et al. (2022b)), and combine it with all executable actions in the environment to enumerate the candidate plans and select the best action with an LM. One difference in these cases is that real-world environments often contain information from multiple modalities, thus requiring multi-modal language models (Li et al., 2019; Lu et al., 2019) that are capable of jointly handling textual, visual, and other modalities.

More concretely, let us consider the task of embodied instruction following, on the popular ALFRED dataset (Shridhar et al., 2020). We use LMs as *high-level planners* for the embodied agent. For example, for a command like *"make me a cup of coffee"*, a high-level plan like [Navigate cup, PickUp cup, Navigate coffee_maker, . . .] is first generated. The agent is equipped with an object detector and a low-level planner to execute the high-level plan from the LM (e.g., navigating to a cup is a classic object localization problem handled by the low-level planner). At each search step, the agent generates a candidate (high-level action, object) pair for each object observed in the environment as possible extensions of the current plan. The LM then scores the candidate expansions similar to KBQA; the best one is executed by the low-level planner.

Preliminary Results. We use the object detector

[6] One necessary step is to convert a SQL query into an algebra tree (Codd, 1970), similar to what is done by Rubin and Berant (2021). In this way, the agent can more easily enumerate the candidate programs in a bottom-up manner. When the database schema is too large, we may define P_0 only as the set of mentioned cell values, then the process of candidate enumeration will resemble KBQA (*i.e.*, cell values can be treated as entities in the KB.)

Composition Rule	Signature	Comments
JOIN	$R \times (E \cup E') \rightarrow E'$	a single hop along an edge
AND	$(T \cup E') \times E' \rightarrow E'$	intersection of two sets
ARGMAX/ARGMIN	$(T \cup E') \times R \rightarrow E'$	superlative aggregations
LT/LE/GT/GE	$R \times E \rightarrow E'$	$< / \leq / > / \geq$
COUNT	$E' \rightarrow N$	set cardinality

Table B.1: Functions in KBQA. We follow the definitions in (Gu and Su, 2022). R: relation, T: type, E: entity, E': a set of entities, N: integer.

and low-level planner from HLSM (Blukis et al., 2022), and use GPT-3.5 `text-davinci-003` as the LM with in-context learning using only 100 labeled examples.[7] We achieved 10% overall success rate and 25% goal completion on ALFRED's unseen dev, already outperforming recent baselines (Pashevich et al., 2021) trained with full data (21K+ examples).

B Candidate Enumeration

Our candidate enumeration for KBQA strictly follows the definition of functions in Table B.1. Specifically, given a set of current plans P_t, to construct the candidate set C_{t+1}, for each plan p_i in P_t, the agent executes it and gets types and relations that are reachable from the denotation of the plan. For each type t, the agent enumerates (AND t p_i) as a candidate. For each relation r, the agent enumerates (JOIN r p_i) as a candidate. If the denotation of p_i is a numerical value, then four similar candidates with comparatives are also included (LT/LE/GT/GE r p_i). In addition, candidate plans with superlatives can be enumerated as (ARGMAX/ARGMIN p_i r). Also, (COUNT p_i) can always be included to C_{t+1}. After checking each p_i independently, the agent then checks each pair of plans p_i and p_j from P_t, if the execution of p_i and p_j has an overlap, then (AND p_i p_j) is also included as a candidate plan. The candidate enumeration process is totally transparent to the LM and can be easily controlled based on different needs.

C Experimental Setup

C.1 Datasets Statistics

All three datasets provide gold program annotations. For consistency, we use the converted S-expressions representation provided by Gu and Su (2022) in our experiments. Concrete statistics of different datasets are shown in Table C.3.

[7] Codex was deprecated on March 23, 2023, so we run our experiments with GPT-3.5 here.

C.2 More Details on Baselines

Different LMs and decoding strategies are used in the baseline models.

ArcaneQA (Gu and Su, 2022) is an encoder-decoder model built on top of a BERT encoder. It leverages constrained decoding and incrementally synthesizes a sequence of subprograms, where the constraints come from both the grammar and the execution of existing subprograms, to enforce grammaticality and faithfulness.

TIARA (Shu et al., 2022) first uses BERT to retrieve a set of schema items, which are further used as the input, together with the question, to T5 for plan generation. They also apply constrained decoding but only for grammaticality.

DecAF (Yu et al., 2022) similarly retrieves a relevant subgraph from the KB using DPR (Karpukhin et al., 2020), and then input the retrieved items to FiD (Izacard and Grave, 2021), a T5 model fine-tuned for question answering.

RnG-KBQA (Ye et al., 2022) first uses BERT to rank a set of enumerated candidate programs (up to a limited complexity), and then uses T5 to edit the top programs into more complex programs.

UnifiedSKG (Xie et al., 2022) also retrieves a subgraph from the KB as input to T5. The setting of UnifiedSKG is different from other baselines. It assumes the gold schema items are always included in the retrieved subgraph and restricts the number of negative schema items in the subgraph (*i.e.*, at most 20 schema items for GRAILQA). It is thus a less fair comparison for other methods, but we include it anyway because it is a representative way of autoregressive plan generation using a large LM.

A summary of the baselines can be found in Table C.2.

C.3 Implementation Details

For GRAILQA we use the entity linking results from TIARA. For WEBQSP, we get that from ELQ (Li et al., 2020), which is also used by our baseline models. For GRAPHQ, get that from ArcaneQA. The entity proposals for the input utterance form the initial plans (P_0) for our search process. We use beam size 5 for all of our fine-tuning experiments. We run our experiments with T5-3B using a single NVIDIA A100 80GB card, while for all other fine-tuning experiments, we run them using $4\times$ NVIDIA A6000 48GB cards.

For our experiments with Codex, we use a beam size of 2 and a max number of candidates of 1,000

Model	LMs	Grounding Strategy	Guarantees
ArcaneQA (Gu and Su, 2022)	BERT-base	Constrained Decoding	Grammatical+Faithful
RnG-KBQA (Ye et al., 2022)	BERT-base + T5-base	Input Augmentation	N/A
TIARA (Shu et al., 2022)	BERT-base + T5-base	Input Augmentation + Constrained Decoding	Grammatical
DecAF (Yu et al., 2022)	DPR + FiD-3B	Input Augmentation	N/A
UnifiedSKG (Xie et al., 2022)	T5-base(/large/3B)	Input Augmentation	N/A

Table C.2: A brief summary of main baseline models.

Dataset	Training	Dev	Test
GRAILQA	44,337	6,763	13,231
GRAPHQ	2,381	–	2,395
WEBQSP	3,098	–	1,639

Table C.3: Statistics of KBQA datasets.

for speed concerns, which to some extent sacrifices the performance. As the first endeavor towards enabling few-shot KBQA with LLMs, we did not tune the hyper-parameters very hard. The only thing we tuned is the scoring function. We tune the scoring function using 10-shot training data from GRAILQA with cross-validation. If we directly use $P(c|u)$ as our scoring function $s(u, c)$ in Section 3.3, Codex tends to favor programs with repeated relations. As a result, we add a penalizing factor to $P(c|u)$, and define $s(u, c)$ as $P(c|u) \times \eta^n$, where $\eta \in [0, 1]$ is a hyper-parameter, and n is the maximal occurrences of a relation in a program. We set $\eta = 0.7$ based on cross-validation using the 10 training examples.

Finally, a small percentage of questions (around 5%) in GRAPHQ and GRAILQA do not have a topic entity (*e.g.*, *"who is the heaviest film director?"* from GRAILQA, whose target program is (ARGMAX film.director people.person.weight_kg)). For these questions, we use the answer types (*e.g.*, film.director) predicted in Gu and Su (2022) as our initial state P_0.

D Decomposition by Question Complexity

We present a fine-grained analysis of Pangu with T5-3B and Codex (100-shot) on questions of different complexity, measured by the number of relations in the gold program, in Table D.4. For GRAILQA, we report the performance on its dev set because the test set is hidden. Pangu performs competitively across all complexity. Note that there are only two questions in GRAILQA's dev set with 4 relations, so the results on that may not be indicative. On GRAPHQ, Pangu significantly outperforms ArcaneQA. The F1 of Pangu with T5-3B is

# of relations	1	2	3	4
RnG-KBQA	79.2	74.8	44.4	**100.0**
ArcaneQA	80.9	71.1	37.7	**100.0**
TIARA	85.6	75.8	**48.5**	83.3
Pangu w/ T5-3B	**87.0**	**78.4**	48.1	83.3
Pangu w/ Codex (100-shot)	73.9	43.4	33.0	16.7

(a) GRAILQA

# of relations	1	2	3
ArcaneQA	48.2	19.3	9.6
Pangu w/ T5-3B	**72.3**	**55.5**	**27.8**
Pangu w/ Codex (100-shot)	52.2	36.1	17.5

(b) GRAPHQ

Table D.4: F1 decomposition by program complexity on GRAILQA's dev set and GRAPHQ's test set.

almost three times higher than ArcaneQA on questions with 2 and 3 relations. Interestingly, Pangu with Codex also outperforms ArcaneQA considerably on questions with 2 and 3 relations. These findings suggest the superiority of Pangu in generalizing to more complex programs.

E Error Analysis

We analyze 200 incorrect predictions (*i.e.*, EM=0) randomly sampled from GRAILQA's dev set for our best model (*i.e.*, T5-3B). The major errors are due to unidentified topic entities during entity linking (62%).[8] Also, Pangu tends to include unrelated entities provided by the entity linker into the final programs (6.5% of the errors), this is because Pangu is fine-tuned with gold entities only, and thus does not learn to handle unrelated entities. In addition, wrong termination check corresponds to 12.5% of the errors, indicating a venue for better enforcing the partial order to Pangu. Apart from these errors, 10.5% of the mistakes are due to ambiguous annotations or annotation errors in GRAILQA. The remaining error types include wrong comparators, answer types, and relations (particularly relations involve a subtle direction like cvg.computer_game_engine.predecessor_engine).

[8]The recall of entity linking on GRAILQA is 88.6% (Shu et al., 2022)

In addition, for in-context learning with Codex (100-shot), we also randomly sample 200 wrong predictions from GRAILQA's dev set. In addition to 22% errors caused by missing entities, the most common errors (25.5%) are due to wrong schema items. Distinguishing gold schema items from confusing ones is challenging for in-context learning. Also, missing constraints (16.5%) and missing relations (10%) are another two major error types, because we use a small batch size (*i.e.*, 2) for Codex and the model tends to prefer short programs. These two error types are also related to wrong termination check. Finally, there are 12% wrong functions. The error types of Pangu w/ Codex are very different from Pangu w/ T5-3B. This is because for a complex task like KBQA, the performance of in-context learning with Pangu still largely lags behind fine-tuning. Particularly, fine-tuning methods directly learn the partial order among programs during training, while Codex needs to implicitly infer a partial order by itself, which is not directly shown in the demonstrations. As a result, Pangu w/ Codex makes more trivial mistakes that fine-tuning methods can easily avoid. More advanced in-context learning techniques to close this gap remains to be explored.

F Examples of Prompts

We show two examples of prompts with 10 in-context samples retrieved from the 100 training data pool in Figure F.1 and Figure F.2 for two different questions from GRAILQA's dev set. Our prompt design is very straightforward. More advanced prompting techniques for Pangu remains to be explored.

```
### Please translate the following questions to Lisp-like query language.

# which automotive designer designed aston martin db7 zagato?
(AND automotive.designer (JOIN automotive.designer.automobiles_designed aston
martin db7 zagato))

# d-series machines was designed by which computer designer?
(AND computer.computer_designer (JOIN
computer.computer_designer.computers_designed d-series machines))

# who designed both visual basic .net and j#?
(AND computer.programming_language_designer (AND (JOIN
computer.programming_language_designer.languages_designed visual basic .net)
(JOIN computer.programming_language_designer.languages_designed j#)))

# which architect designed katherine atkins house by polk?
(AND architecture.architect (JOIN architecture.architect.structures_designed
katherine atkins house by polk))

# what is the name of the author who wrote it is an open question whether any
behavior based on fear of eternal punishment can be regarded as ethical or should
be regarded as merely cowardly.?
(AND film.director (JOIN media_common.quotation.author_inv it is an open question
whether any behavior based on fear of eternal punishment can be regarded as
ethical or should be regarded as merely cowardly.))

# who was the manufacturer of kosmos 3m?
(AND spaceflight.rocket_manufacturer (JOIN
spaceflight.rocket_manufacturer.rockets_manufactured kosmos 3m))

# who is the endorser of coke products?
(AND business.product_endorser (JOIN business.product_endorsement.endorser_inv
(JOIN business.product_endorsement.product coke)))

# what short story has a character who also is in doing clarence a bit of good?
(AND book.short_story (JOIN book.short_story.characters (JOIN
book.book_character.appears_in_stories doing clarence a bit of good)))

# who was the director of the episode kate jackson/delbert mcclinton?
(AND tv.tv_director (JOIN tv.tv_director.episodes_directed kate jackson/delbert
mcclinton))

# what is the identity of the football player who appeared 23 times
internationally?
(AND soccer.football_player (JOIN
soccer.football_player.total_international_appearances 23))

# what is the role of opera designer gig who designed the telephone / the medium?
```

Figure F.1: Example prompt (i) for question *"what is the role of opera designer gig who designed the telephone / the medium?"*

```
### Please translate the following questions to Lisp-like query language.

# homegrown is a recurring segment on what tv program?
(AND tv.tv_program (JOIN tv.tv_program.recurring_segments homegrown))

# on 07/01/1970, which warship v1.1 was hit?
(AND user.patrick.default_domain.warship_v1_1 (JOIN
user.patrick.default_domain.warship_v1_1.struck 07/01/1970))

# what is the isbn of the edition with scott fisher on its book cover?
(AND book.isbn (JOIN book.book_edition.isbn_inv (JOIN
book.illustrator.book_edition_covers_inv scott fisher)))

# which musical artist stopped being active as musical artist on 1985-06?
(AND music.artist (JOIN music.artist.active_end 1985-06))

# the honorary degree recipient that was born most recently is named what?
(ARGMAX education.honorary_degree_recipient people.person.date_of_birth)

# the medical trials conducted on safety and effectiveness of giving indinavir
plus stavudine plus lamivudine to hiv-infected children are under the authority
of who?
(AND medicine.medical_trial_health_authority (JOIN
medicine.medical_trial_health_authority.medical_trials safety and effectiveness
of giving indinavir plus stavudine plus lamivudine to hiv- infected children))

# bataan 1 and bataan 2 is what aircraft model?
(AND aviation.aircraft_model (JOIN aviation.aircraft_model.aircraft bataan 1 and
bataan 2))

# what ingredient is in french cuisine?
(AND food.ingredient (JOIN food.ingredient.cuisine french cuisine))

# south kent school and redfield college fall under what category of school?
(AND education.school_category (AND (JOIN
education.educational_institution.school_type_inv south kent school) (JOIN
education.school_category.schools_of_this_kind redfield college)))

# chiang kai shek college and sacred heart high school (roseville, michigan) are
in what category of school? (AND education.school_category
(AND (JOIN education.educational_institution.school_type_inv chiang kai shek
college) ( JOIN education.school_category.schools_of_this_kind sacred heart high
school (roseville, michigan))))

# semaphore railway line is on the rail network named what?
```

Figure F.2: Example prompt (ii) for question *“semaphore railway line is on the rail network named what?"*

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
The section after conclusion, following the instruction in the Latex file.

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4.1; Section C.1

☑ B1. Did you cite the creators of artifacts you used?
Section 4.1; Section C.1

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 4.1; Section C.1

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The used datasets are all widely used for this task and there is no report of identifying or offensive content as far as we know

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4.1; Section C.1

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section C.1

C ☑ Did you run computational experiments?

Section 5.1; Section 5.2; Section D

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4.3; Section C.3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section C.3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5.1; Section 5.2

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.3; Section C.3

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.