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 ability 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 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 candidate 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 achieving a new state of the art on standard KBQA datasets, and larger LMs further improve the performance by a large margin. Pangu also enables, for the first time, effective few-shot in-context learning for KBQA with large LMs such as Codex.[1]

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 ability 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[2] 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.

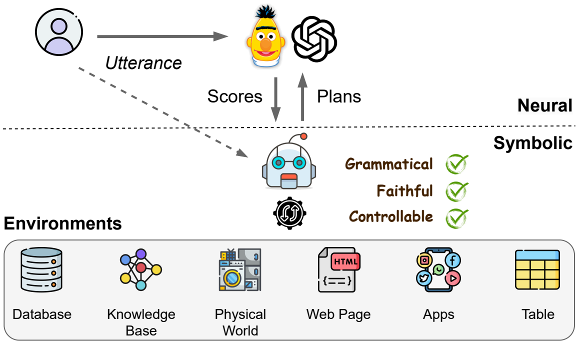

Figure 1: A schematic illustration of the proposed framework, Pangu, where a symbolic agent interacts with the target environment to propose candidate plans, and a neural LM evaluates the plausibility of each plan based on the input utterance. The agent searches in the environment to incrementally construct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

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 in household settings (Shridhar et al., 2020)). 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 in a large database of electronic medical records who exhibited a rare combination of symptoms 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*

[1] Code and data will be released at https://github.com/dki-lab/Pangu.

[2] chat.openai.com

natural language?

Each environment is a unique context for interpreting natural language requests from users. *Grounding*, *i.e.*, linking of (natural language) concepts 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 a left-to-right autoregressive fashion (Wang et al., 2021; Xie et al., 2022; Ye et al., 2022; Liu 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 specific environment to ensure the *grammaticality* (*i.e.*, conforming to the grammar of the planning language) and *faithfulness* (*i.e.*, executable in the environment) of the generated plans, neither of which may be part of an LM's pre-training. 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. The focus of recent work is to alleviate (some of) these three limitations by augmenting the autoregressive generation paradigm 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 or sample the target plan. It thus casts the burden of ensuring grammaticality, faithfulness, and controllability all on the LM itself. The main proposal of this paper is to *disentangle LMs from these responsibilities and let LMs be what they are originally defined as—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,[3] a generic framework for grounded language understanding that capitalizes on the discriminative ability of LMs instead of their generative ability (Figure 1). Pangu consists of a symbolic agent and a neural LM working in a concerted way. The symbolic agent operates in the environment to propose candidate plans, which are guaranteed by design to be both grammatical and faithful. If the environment is tiny, it may be possible to enumerate all candidate plans up front and let the LM select the best one. For most realistic environments, however, 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 for the next step. The LM also determines when the search ends, *i.e.*, when no further extension of the current best plan would score higher. Finally, it is much easier to control the search process of a symbolic agent than to control the generation process of a neural LM. For example, one can easily define a list of disallowed actions for a certain search state and prevent the agent from proposing such plans.

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 understand-

[3] Pangu is a primordial being in Chinese mythology who separated heaven and earth. We name our framework after Pangu for its separating the realm of the neural (*i.e.*, the neural LM) and the symbolic (*i.e.*, the symbolic agent).

ing because of its massive environment. We show that simply using BERT-base with Pangu is sufficient for achieving a new state of the art on standard KBQA benchmarks, and using larger LMs further improves the performance by a large margin. Pangu also enables, for the first time, prompting large language models (*e.g.*, Codex) for few-shot KBQA with competitive performance. It provides *unprecedented uniformity* for using LMs: one can easily plug encoder-only LMs (*e.g.*, BERT), encoder-decoder LMs (*e.g.*, T5 (Raffel et al., 2020)), or decoder-only LMs (*e.g.*, Codex) into Pangu. These results highlight the remarkable effectiveness and flexibility of Pangu and validate the proposal of using LMs for discrimination instead of generation for grounded language understanding.

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 over the environment given an input utterance. However, due to the lack of grounding during pre-training, generating valid plans from the LM is challenging. Recent studies endeavor to alleviate this issue via *inference-time augmentation* or *pre-training augmentation*. Two families of inference-time augmentation have been proposed so far, namely, *input augmentation* and *constrained decoding*. For input augmentation, the environment—or a representative subset of the environment—is fed to the LM's encoder together with the utterance (Hwang et al., 2019; Xie et al., 2022). Such methods rely on the LM itself to understand the environment and output a plan, and thus are more data-inefficient and have no guarantee for grammaticality or faithfulness. In contrast, several recent methods use constrained decoding to regulate the decoder's behavior, which can 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 grammatical and faithful plans on the LM itself. In our proposal, the LM is only used to discriminate candidate plans proposed by an agent that explores the environment.

Different from inference-time augmentation, pre-training augmentation methods seek to reduce the gap between LMs' pre-training settings and grounded language understanding tasks by performing environment-specific pre-training (Yu et al., 2021; Deng et al., 2021). However, these methods need a large set of environment-specific data for pre-training (*e.g.*, aligned corpus for table schema and utterances (Yu et al., 2021)) and do not generalize to different environments (*e.g.*, LMs with table-specific pre-training are not applicable to knowledge bases). By contrast, Pangu requires no pre-training and can be easily plugged into different environments.

2.2 Few-Shot Grounded Language Understanding with LLMs

The use of Large Language Models (LLMs) such as GPT-3 (Brown et al., 2020) and Codex (Chen et al., 2021a) in grounded language understanding tasks has attracted increasing interest recently. LLMs have demonstrated strong few-shot learning capabilities in a variety of environments, from writing programs to query and manipulate structured and unstructured data sources (Austin et al., 2021; Rajkumar et al., 2022; Cheng et al., 2022), interacting with mobile UI (Wang et al., 2022) and 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., 2022). Most existing work still capitalize on the generative ability of LLMs. A common strategy to encourage an LLM to produce valid plans is to directly *describe* the environment as context to the LLM (*i.e.*, input augmentation), which is difficult for more complex environments like KBs. 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. Our framework 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 sub-components. At each step of

Algorithm 1: PANGU

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

parsing, SmBoP enumerates candidate parse trees (*i.e.*, plans) from all valid combinations of trees from the previous step and scores them to get the top-ranked ones. Pangu similarly constructs a complex plan incrementally from smaller subplans, but makes the following three 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 implementation, only topic entities are required at the beginning, 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. Finally, SmBoP parses for fixed T steps, while Pangu adaptively terminates the search process based on the scores from the LM. Also related is an array of earlier KBQA methods that adopt an enumerate-and-rank approach (Yih et al., 2015; Chen et al., 2019; Lan and Jiang, 2020). Because they try to enumerate all candidate plans up front, the maximum complexity of the plans is bound to be small. Our adaptive search process allows for flexible generation of more complex plans.

3 Approach

3.1 The Pangu Framework: Overview

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/knowledge base (KB) (Yu et al., 2018; Gu et al., 2022). Starting from a set of initial plans P_0 (maybe 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 parameter-free symbolic agent explores the environment to propose valid plans and shields the LM from having to handle the large search space for plan generation. Instead, the LM only focuses on evaluating the plausibility of the proposed plans, *i.e.*, to what extent a candidate plan matches the intent of the input utterance. 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.

3.2 KBQA: Preliminaries

Without loss of generality, we use KBs as our target environment and the knowledge base question answering (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 plan/program[4] 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.

3.3 Candidate Plan Enumeration

To handle the large search space, the agent casts the task as a step-wise decision-making problem. For example, an instruction-following robot may

[4] we will use plan and program interchangeably hereafter.

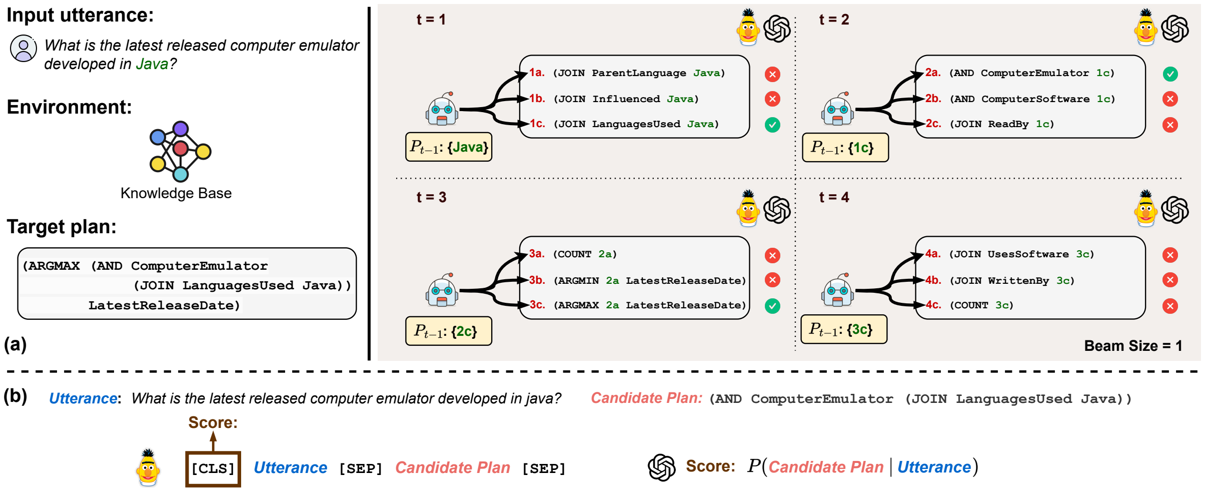
Input utterance:
What is the latest released computer emulator developed in Java?
Environment:
Knowledge Base
Target plan:
(ARGMAX (AND ComputerEmulator (JOIN LanguagesUsed Java)) LatestReleaseDate)
(a)
t = 1
1a. (JOIN ParentLanguage Java)
1b. (JOIN Influenced Java)
1c. (JOIN LanguagesUsed Java)
P_{t-1}: {Java}
t = 2
2a. (AND ComputerEmulator 1c)
2b. (AND ComputerSoftware 1c)
2c. (JOIN ReadBy 1c)
P_{t-1}: {1c}
t = 3
3a. (COUNT 2a)
3b. (ARGMIN 2a LatestReleaseDate)
3c. (ARGMAX 2a LatestReleaseDate)
P_{t-1}: {2c}
t = 4
4a. (JOIN UsesSoftware 3c)
4b. (JOIN WrittenBy 3c)
4c. (COUNT 3c)
P_{t-1}: {3c}
Beam Size = 1
(b)
Utterance: What is the latest released computer emulator developed in java?
Candidate Plan: (AND ComputerEmulator (JOIN LanguagesUsed Java))
Score:
[CLS] Utterance [SEP] Candidate Plan [SEP]
Score: $P($Candidate Plan | Utterance$)$

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 enumerates a set of valid plans based on the current plans and the environment. 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.

decompose a plan 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. (2022)). Similarly, a program for KBQA can be decomposed into a nested sequence of subprograms (Rubin and Berant, 2021; Gu and Su, 2022) (Figure 2). The *length* of a plan is defined as the number of atomic subplans it contains.

For KBQA (and similar semantic parsing tasks), 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 A for a more detailed discussion of this process.

3.4 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 sentence-pair classification: given a pair of (u: *utterance*, $c \in C_t$: *candidate plan*). An LM behaves as a scoring function: $S(u, c) \rightarrow \mathbb{R}$, which indicates to what extent the candidate plan matches the intent of the utterance. Evaluating the plausibility of each candidate basically boils down to *semantic matching* based on linguistic cues. For example, an LM should be able to tell that $2a$ in Figure 2 is a positive candidate due to the keyword ComputerEmulator in its surface form.

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, we follow Zhuang et al. (2022) to feed both the utterance and the candidate plan to the encoder and use the decoding probability over an unused token[5] during pre-training as a proxy for matching score. 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 should re-

[5] We use <extra_id_23> for T5.

spect the following *partial order*:

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

where G_t is the set of gold (sub-)plans at step t (*i.e.*, length-t subplans of the target plan) 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 negative plans 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, the gold target plan should be scored higher than any other plan.

3.5 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 candidate plan of length $\geq t$ is more complete than the plans at step $t-1$, and thus the search process terminates. It is worth noting that this may not be the only way of checking termination for other grounded language understanding tasks. For example, an instruction-following robot may check the environment state (*e.g.*, whether a cup of coffee has been successfully made) for that purpose.

3.6 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.

3.6.1 Fine-Tuning

Given a gold plan of length T, we first derive its gold sub-plans G_t of each step $t \leq T$ (*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 following 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).

3.6.2 In-Context Learning

For in-context learning, we directly use pairs of utterances and gold plans as in-context examples 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 B.

4 Experimental Setup

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

Table 1: Statistics of KBQA datasets.

4.1 Datasets

We experiment with three standard KBQA datasets of different scale and nature (Table 1).

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 that focuses on non-i.i.d. generalization in KBQA. Due to the small size of its training set and the non-i.i.d. setting, GRAPHQ is a particularly challenging benchmark. In our experiments, we use the processed version by Gu and Su (2022),

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 2: A brief summary of main baseline models.

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.

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.

4.2 Baselines

We mainly compare Pangu with multiple state-of-the-art baselines that use LMs as a generative model for KBQA. 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.

There is no constrained decoding for the last three baseline methods, and thus neither grammaticality nor faithfulness can be guaranteed. These methods rely on labeled training data to learn how to produce valid programs; their sample efficiency suffers as a result (§5.3). A summary of the baselines can be found in Table 2. Compared with the baselines, Pangu requires no extra parameter, no modification to the LM, and no need to combine multiple LMs (for retrieval/encoding/decoding). Pangu also provides unprecedented uniformity and flexibility of using LMs, and one can easily plug LMs of different nature, *e.g.*, encoder-only, encoder-decoder, or decoder-only, into our framework and they work in a plug-and-play fashion.

4.3 Implementation Details

We instantiate Pangu for KBQA. 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.[6] We first randomly sample 100 training examples from each dataset and use that as the pool for dynamic retrieval, therefore a *100-shot* setting. During inference, for each test example, we retrieve 10 in-context examples from the pool using BM25-based utterance similarity.

We use entity linking results from off-the-shelf entity linkers. 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 our fine-tuning experiments and beam size 1 for Codex. We run our experiments with T5-3B using a single NVIDIA

[6] We opt for Codex because it is free, but small-scale experiments also show competitive performance from GPT-3.

	Overall		I.I.D.		Compositional		Zero-shot		Dev Overall	
Model	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**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., 2022b)	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 (Xie et al., 2022)										
w/ T5-3B	–	–	–	–	–	–	–	–	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 (100-shot)	46.1	53.3	51.4	57.2	46.5	54.0	43.4	51.2	47.3	53.5

(a) GrailQA

Model	**F1**
UDEPLAMBDA (Reddy et al., 2017)	17.7
PARA4QA (Dong et al., 2017)	20.4
SPARQA (Sun et al., 2020)	21.5
BERT+Ranking (Gu et al., 2021)	25.0 (27.0)
ArcaneQA (Gu and Su, 2022)	31.8 (34.3)
Pangu (this work)	
w/ BERT-base	48.2 (52.0)
w/ T5-base	49.4 (53.3)
w/ T5-large	51.6 (55.6)
w/ T5-3B	**57.7 (62.2)**
w/ Codex (100-shot)	32.8 (35.4)

(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., 2022b)	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 (100-shot)	48.6

(c) WebQSP

Table 3: Overall results on three KBQA datasets. For GraphQ, numbers in parenthesis indicate results excluding the unconvertible questions in GraphQ 2013-07 following Gu and Su (2022). 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. * denotes using oracle entity linking.

A100 80GB card, while for all other fine-tuning experiments, we run them using 4× NVIDIA A6000 48GB cards.

5 Results

5.1 Main Results

Fine-tuning results. The main results are shown in Table 3. 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 GrailQA, Pangu with BERT-base outperforms the prior art DecAF by 5.3% in EM and 1.2% in

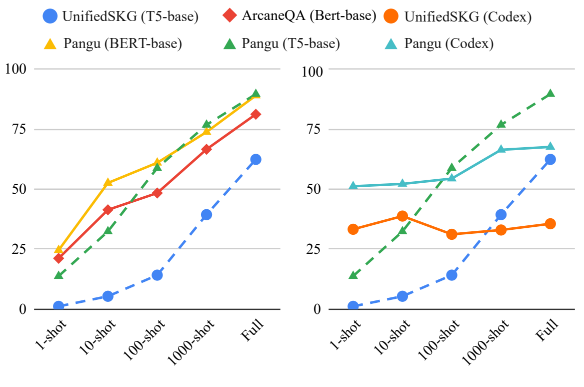

Figure 3: Sample efficiency results. *Left*: fine-tuning, *right*: in-context learning.

# 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	64.6	33.5	19.5	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	46.3	24.8	11.1

(b) GRAPHQ

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

F1. 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, *i.e.*, Pangu with BERT-base improves the previous best zero-shot F1 by 2.2%. Because GRAILQA has more training data, which gives models a better transfer learning opportunity for zero-shot generalization, our gain is not as conspicuous as on GRAPHQ but still decent. Our method also shows great flexibility in accommodating different LMs 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 easier for LMs to learn than the generative task, Pangu converges very fast and gets fewer training steps for overfitting the i.i.d. setting, in exchange for better non-i.i.d. generalization. For both BERT and T5, Pangu takes at most two epochs to converge. The strong performance on WEBQSP, an i.i.d. dataset, further supports this observation.

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 53.3%. Though there is still a gap to the fine-tuning results, it is still impressive given that only 10 in-context examples and 100 training examples in total are used, especially considering the massive meaning space of the KB. On GRAPHQ, Pangu with Codex even outperforms ArcaneQA. This finding is consistent with our fine-tuning experiments, *i.e.*, Pangu is particularly strong in generalizing to new environments with limited training data. On WEBQSP, Pangu trails behind fine-tuning methods, which is expected given WEBQSP's i.i.d. nature. Fine-tuning methods can better memorize the patterns in the training data.

5.2 Decomposition by Question Complexity

We present a fine-grained analysis of Pangu with T5-3B and Codex on questions of different complexity, measured by the number of relations in the gold program, in Table 4. For GRAILQA, we report the performance on its dev set because the test set is hidden. Unsurprisingly, performance generally decreases as question complexity increases. 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 almost three times higher than ArcaneQA on questions with 2 and 3 relations. Interestingly, Pangu with Codex (100-shot) also outperforms ArcaneQA considerably on questions with 2 and 3 relations. These findings suggest the superiority of Pangu in generalizing to more complex programs, but also leave room for further improvement, especially for in-context learning.

5.3 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. We also adapt UnifiedSKG to work under in-context learning with Codex, so we can have a direct comparison between discrimination vs. generation under in-context learning. First, we observe that when both using T5-base, UnifiedSKG significantly underperforms Pangu. The main reason is that most predicted plans are ungrammatical or unfaithful in the low-data regimes. ArcaneQA uses constrained decoding to alleviate this issue, but still consistently underperforms Pangu when both using BERT-base. We also observe that T5-base performs worse than BERT-base under 1-shot and 10-shot settings, which can be explained by the added complexity of T5 that makes it harder to train under extreme low-data settings.

For in-context learning using Codex, Pangu achieves an EM of over 50% with only one training instance, which significantly outperforms all other settings, including UnifiedSKG+Codex. This may point to interesting opportunities for practical KBQA under extreme low-data settings.

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, for grounded language understanding. It 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. As a case study, we instantiated the proposed framework for the challenging knowledge base question answering (KBQA) task, which features a massive environment and a particularly challenging setting for grounding. Extensive experiments convincingly demonstrated the remarkable effectiveness and flexibility of the proposed framework. 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 for future improvement. The first major limitation lies in efficiency. Because Pangu requires an LM to iteratively score candidate plans, it tends to be more resource-consuming, in terms of both time and computing. Compared with ArcaneQA, which efficiently handles complex questions in KBQA, Pangu is about twice slower for both training and inference and consumes about twice as much GPU memory when both using BERT-base. Specifically, 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. To alleviate this issue, algorithms with a complexity better than $O(N)$, N is the number of candidate plans, can be developed to find the top-K candidates.

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 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 (Wei et al., 2022).

Last but not least, 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).

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 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. *arXiv preprint arXiv: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.

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.

Zi-Yuan Chen, Chih-Hung Chang, Yi-Pei Chen, Jijnasa Nayak, and Lun-Wei Ku. 2019. UHop: An unrestricted-hop relation extraction framework for knowledge-based question answering. 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 345–356, Minneapolis, Minnesota. 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.

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.

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.

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.

Haoyu Liu, Yang Liu, Hongkai He, and Hangfang Yang. 2022a. LEBP - language expectation & binding policy: A two-stream framework for embodied vision-and-language interaction task learning agents. *CoRR*, abs/2203.04637.

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

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.

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. Computer Vision Foundation / IEEE.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge bases. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

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, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022. 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.

Bryan Wang, Gang Li, and Yang Li. 2022. Enabling conversational interaction with mobile UI using large language models. *CoRR*, abs/2209.08655.

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.

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

Table 5: 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.

A Candidate Enumeration

Our candidate enumeration for KBQA strictly follow the definitions of different functions in Table 5. 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 its execution by exploring the KB. For each type t, the agent enumerates (AND t p_i) as a candidate. For each relation r, the agent enumerate (JOIN r p_i) as a candidate. If the execution 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 check each p_i independently, the agent then check 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.

B Examples of Prompts

We show two examples of prompts with 10 in-context samples retrieved from the 100 training data pool in Figure 4 and Figure 5.

```
### Please translate the follow 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 4: Example Prompt 1.

```
### Please translate the follow 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 5: Example Prompt 2.