Extrapolating AI Agents to Open-World Tasks with Natural Language

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Yu Gu, B.Sc., M.Sc.

Graduate Program in Department of Computer Science and Engineering

The Ohio State University

2025

Dissertation Committee:

Dr. Yu Su, Advisor

Dr. Huan Sun

Dr. Wei-Lun Chao

Dr. Zhihui Zhu

Dr. Jonathan Berant

Abstract

Enabling machines to take actions and fulfill diverse human goals constitutes the most long-standing objective of Artificial Intelligence (AI) research. This thesis addresses the problem of developing AI agents in *open-world* environments characterized by unbounded states, actions, and broad goal specifications—the ultimate objective in AI agent development.

A core challenge for developing such versatile agents in open-world tasks lies in effective *goal transfer* (*i.e.*, extrapolating to diverse unseen goals): Classic AI agents operate exclusively within structured logic worlds, while mapping environment dynamics and goals into logic symbols in open-world tasks is extremely challenging, if not impossible; Agents based on reinforcement learning (RL) specify goals as reward functions and are typically only optimized to achieving a *fixed* goal (*e.g.*, playing Go or cleaning a table). Extrapolating to new goals would require specifying new reward functions and optimization against them.

We advocate for explicitly modeling goals in natural language in AI agents. This not only creates a friendly interface between humans and agents, but more importantly, it also allows agents to extrapolate to new goals more easily: imagine an agent that can *"water plants in the kitchen"* and *"fold laundry in the bedroom"*—ideally, the agent should understand and execute *"water plants in the bedroom"* by composing these concepts without requiring additional training for this new goal. Particularly, there are two core components that contribute to extrapolation with natural language modeling: 1) Natural language modeling produces a structure for the goal space, including similarity

measurements, compositions, *etc.* 2) Prior knowledge encoded in natural language can facilitate the mapping from goal to policy.

We detail our discussions in two parts: Part I focuses on providing insights into how modeling natural language goals helps in extrapolation. Specifically, leveraging the knowledge base (KB) environment as a testbed, we first establish a benchmark that systematically defines extrapolating to new goals at three levels: *i.i.d.*, *compositional*, and *zero-shot* extrapolation, which are determined based on the relationship between the training data and test data. Our evaluation using this benchmark shows the critical role of pre-trained language models in goal modeling and of constrained decoding and discrimination in mapping goals to actions. These findings are further reinforced by two follow-up methods that enhance extrapolation within the KB environment.

However, agents in the KB environment operate with a strong assumption that the goal description and available actions are of a similar abstraction level. This strong assumption does not always hold. For example, in a household setting, a robot instructed to *"make a pizza"* probably does not come with an atomic action called `make_pizza`. To handle this, we go one step further in Part II, in which we investigate ways to leverage prior knowledge encoded in large language models (LLMs) to boost the mapping from goal to policy, without using any training data. We discuss two works. First, we introduce how to use natural language prior to enhance policy greedily in the KB and database environments. Second, we shift to the more complex real-world website environment and show that we can also use natural language prior for model-based planning, *i.e.*, using LLMs to simulate a world model for the Internet and conduct systematic exploration within the simulation.

With the work presented in this thesis, we demonstrate the critical role of natural language modeling in developing extrapolative agents in open-world environments. Beyond serving as a communication interface, natural language enables effective modeling of the goal space and provides a prior for the mapping between goals and policies. However, this thesis is just a beginning, more

future breakthroughs to incorporate natural language capability more seamlessly into the RL or classic agent framework are yet to come. Finally, the more generalizable statement of this thesis is that policies should be grounded in a concept space (*i.e.*, *concept-grounded policies*), whether derived from language learning or not.[1] The fundamental challenge in AI research remains how to effectively form concept spaces that mirror human cognitive structures.

[1]This might sound conflicting with the previous statement, but it is not. The reality is that currently we may not have a better way to derive human concepts than directly learning from massive language corpora.

Dedicated to my family, my advisor, my friends, and myself—whose collective support, guidance, and perseverance made this journey possible.

Acknowledgments

Back in 2019, I made a major life decision—to accept the research assistantship offer from Dr. Yu Su. He later became my PhD advisor, and I had the distinction of being his first doctoral student. I still have the email he sent after I accepted the offer:

> *"Pursuing a PhD degree is truly a major life decision. It could be both extremely frustrating and extremely rewarding. Let's work closely together to make your best few years count and make a real impact!"*
>
> ---
>
> *Email from Dr. Yu Su, February 6, 2019*

As it turns out, Dr. Su was absolutely right—this journey has been full of ups and downs, with both rewarding and frustrating moments. Also, it turns out these few years have indeed become my "best few years"—I've grown faster than ever before, in many aspects of my life. Not only did I acquire the essential skills needed to become an independent researcher, but also developed a growth mindset that views all challenges and obstacles as opportunities to grow, making me more resilient under various circumstances. This transformation would not have been possible without proper mentorship. For this reason, I would like to thank my advisor, Dr. Su, first and foremost, for his unwavering support and invaluable guidance throughout all the rewarding and frustrating moments.

In addition to my advisor, I have been fortunate to receive help from many wonderful people throughout this journey. Dr. Huan Sun, who has been another important mentor during my PhD, consistently offered valuable career advice and emotional support. Working with her has been a truly rewarding experience—one that I will deeply miss in the future. I am also grateful for the guidance

and encouragement of several senior researchers: Dr. Gong Cheng, my advisor during my studies at Nanjing University, as well as Dr. Jie Tang and Dr. Yuxiao Dong, who graciously hosted me during my visit to Tsinghua University. Each of them has been exceptionally generous with their time and support, for which I am deeply grateful. Furthermore, I am honored to have had such an outstanding group of committee members on this journey: Dr. Jonathan Berant, Dr. Wei-Lun Chao, Dr. Zhihui Zhu, and Dr. Alexander Petrov, in addition to Dr. Su and Dr. Sun. Their insightful feedback has been important to the completion of my thesis. Last but not least, I would also like to thank my senior collaborators—Percy Liang, Brian Sadler, Xifeng Yan, and Jiawei Han—for their thoughtful advice and mentorship.

Besides the help from the senior researchers, I am also deeply grateful to many of my peers—several of whom have become not only important collaborators but also close friends. A shared passion for research brought us together, and I truly value the connections and experiences we have shared. There are so many of them that I sincerely hope I don't miss anyone: Bernal Jiménez, Vardaan Pahuja, Chan Hee (Luke) Song, Zhen Wang, Ziyu Yao, Jie Zhao, Xiang Deng, Xiang Yue, Lingbo Mo, Yiqi Tang, Xinliang (Frederick) Zhang, Bernhard Kratzwald, Shijie Chen, Sam Stevens, Kai Zhang, Jiaman (Lisa) Wu, Boyuan Zheng, Yiheng Shu, Boyu Gou, Zanming Huang, Jianyang Gu, Boshi Wang, Zeyi Liao, Yuting Ning, Tianci Xue, Michael Lin, Eddy Luo, Zishuo Zheng, Tianshu Zhang, Ron Chen, Yifei Li, Botao Yu, Jaylen Jones, Harsh Kohli, Weijian Qi, Jian Xie—those from the OSU NLP group—and my friends & collaborators beyond the lab: Zihao Zhou, Yuntian He, Dehan Kong, Yaodong Yu, Kun Tao, Zixian Huang, Wentao Ding, Qiaosheng Chen, Xiaxia Wang, Yuxin Wang, Xiao Liu (THU), Tianjie Zhang, Hao Yu, Yifan Xu, Hanyu Lai, Zhenyu Li, Sunqi Fan, Shulin Cao, Lunyiu Nie, Jifan Yu, Xiaozhi Wang, Ming Ding, Wenyi Hong, Jiazheng Xu, Wendi Zheng, Zhuoyi Yang, Jiayan Teng, Bo Chen, Qinkai Zheng, Boyan Wang, Wei Wang, Dan Zhang, Yujia Qin, Chenyang Zhao, Wei Liu, Shudong Liu, Tianle Li, Qian Liu, Zhiwei Yu, Shuang

Chen, Hanjie Chen, Yaqing Wang, Zhoujun Cheng, Fan Zhou, Junlong Li, Da Yin, Xiao Liu (PKU), Donghan Yu, Xi Ye, Jiaxian Guo, Wenyue Hua, Shuyan Zhou, Yongchao Chen, Michihiro Yasunaga, Chenglei Si, Xidong Feng, Ritam Dutt, and Dongfang Lin. I feel incredibly fortunate to have had the opportunity to collaborate with or engage in meaningful discussions with each of you.

I would also like to thank my girlfriend, WZ, for her companionship throughout this journey. Her presence has been a constant source of strength and comfort. In addition, I am also grateful for my dog and cat, GXW and SXM, who have brought me so much emotional support and endless joy. Finally, I owe my deepest and most sincere gratitude to my parents for their endless love. My parents have always been my biggest supporters—unconditionally believing in me and never questioning my choice. Their steady support has allowed me to focus wholeheartedly on my goals, free from any distraction or worry.

The end of this journey marks not a conclusion, but the beginning of a new chapter. I look forward to carrying the lessons, friendships, and inspiration from this time into whatever comes next—and I hope to continue growing, learning, and contributing alongside the incredible people I've been fortunate to meet along the way.

Vita

September 2012 - June 2016	B.Sc., Department of Computer Science and Technology, Nanjing University, Nanjing, China.
September 2016 - June 2019	M.Sc., Department of Computer Science and Technology, Nanjing University, Nanjing, China.
August 2019 - April 2025	Ph.D. Student, Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA.
August 2019 - April 2025	Graduate Research Associate, Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA.
May 2021 - August 2021	Research Intern, Microsoft Research, Redmond.

Selected Awards:

2022	Outstanding Paper Award, COLING
2023	Outstanding Paper Award, ACL
2024	Outstanding Reviewer, EMNLP
2025	Graduate Student Research Award, OSU CSE

Selected Publications

(* indicates equal contribution)

Boyuan Zheng*, Michael Y. Fatemi*, Xiaolong Jin*, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, **Yu Gu**, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *CoRR*, abs/2504.07079, 2025

Yu Gu*, Kai Zhang*, Yuting Ning*, Boyuan Zheng*, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *CoRR*, abs/2411.06559, 2025

Xiao Liu*, Tianjie Zhang*, **Yu Gu***, Iat Long Iong, Xixuan Song, Yifan Xu, Shudan Zhang, Hanyu Lai, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. Visualagentbench: Towards large multimodal models as visual foundation agents. In *The Thirteenth International Conference on Learning Representations*, 2025

Bernal Jiménez Gutiérrez, Yiheng Shu, **Yu Gu**, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. Middleware for llms: Tools are instrumental for language agents in complex environments. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7646–7663, 2024

Xiao Liu*, Hao Yu*, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, **Yu Gu**, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*, 2024

Zhenyu Li, Sunqi Fan, **Yu Gu**, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18608–18616, 2024

Xiang Deng, **Yu Gu**, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023

Donghan Yu, **Yu Gu**, Chenyan Xiong, and Yiming Yang. Compleqa: Benchmarking the impacts of knowledge graph completion methods on question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12748–12755, 2023

Tianle Li, Xueguang Ma, Alex Zhuang, **Yu Gu**, Yu Su, and Wenhu Chen. Few-shot in-context learning on knowledge base question answering. In *Annual Meeting of the Association for Computational Linguistics*, 2023

Yu Gu, Xiang Deng, and Yu Su. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4928–4949. Association for Computational Linguistics, 2023

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

Zixian Huang, Ao Wu, Jiaying Zhou, **Yu Gu**, Yue Zhao, and Gong Cheng. Clues before answers: Generation-enhanced multiple-choice qa. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3272–3287, 2022

Yu Gu and Yu Su. 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, October 2022. International Committee on Computational Linguistics

Vardaan Pahuja, **Yu Gu**, Wenhu Chen, Mehdi Bahrami, Lei Liu, Wei-Peng Chen, and Yu Su. A systematic investigation of kb-text embedding alignment at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1764–1774, 2021

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

Fields of Study

Major Field: Department of Computer Science and Engineering

Studies in:

Artificial Intelligence	Dr. Yu Su
Data Mining	Dr. Huan Sun
Probabilistic Models	Dr. Yingbin Liang

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Enabling machines to comprehend various goals provided by humans and correspondingly take actions has been the most long-standing goal in Artificial Intelligence (AI) research (Figure 1.1).

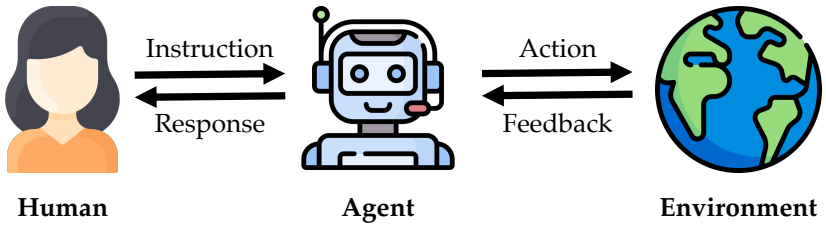

Figure 1.1: A high-level illustration of the concept of an AI agent. The agent accomplishes goals specified by a human by interacting with a target environment, optionally with a human in the loop.

Research on AI agents starts with rule-based methods with hand-crafted code to define agent behavior (*e.g.*, Xcon [105], GPS [113], and DeepBlue [20]). This reliance on manual effort makes it difficult to handle complex real-world environments. As a result, early research has focused on highly specialized domains, such as computer system configuration [105], and simple game environments whose dynamics are guided by a small set of rules, such as chess [20] or puzzles [113]. More recently, AI agent research has shifted towards reinforcement learning (RL), where the agent learns its policy through trial and error by interacting with the environment. Compared with rule-based methods, the RL-based approach enables agents to handle more dynamic and complex environments, such

as Atari games [109], as they can self-evolve based on accumulated experience rather than relying solely on pre-defined rules. RL has also led to major breakthroughs, such as achieving superhuman performance in Go (*i.e.*, AlphaGo [152]). However, a critical gap is that much of the existing research, whether rule-based or RL-based, centers on agents with narrowly defined objectives (*e.g.*, game playing). This thesis aims to narrow this gap and focus on how to better develop agents that can serve diverse goals in *open-world tasks*. Presumably, AI agents capable of serving open-world tasks may become one of humanity's most significant technological innovations, with unprecedented autonomy and flexibility.

1.1 Open-World Tasks

Although there may be no universal agreement on the definition of open-world tasks [187], this thesis will try to establish a clear scope of our focus. In particular, this thesis mainly uses the term open-world tasks to contrast with typical settings featured in classic logic-based agents or agents based on RL with bounded states, actions, and fixed goals. In classic logic-based agents, both environmental dynamics and goals should be modeled using logic symbols, and this modeling is usually done with the manual effort of human experts (*e.g.*, Xcon [105], GPS [113], and DeepBlue [20]). RL agents may relax the constraint on modeling environment dynamics; they typically focus only on fixed goals specified through reward functions (*e.g.*, playing Go [152]).[2] Note that, there are also RL environments support various goals, such as navigating to different locations [3] or manipulating different objects on a table [188]. However, these tasks typically operate within a constrained environment (*e.g.*, a constrained table manipulation environment) and still feature highly limited action space and homogeneous goal space (*e.g.*, goals may only differ in the target location to reach or the target object to manipulate). Additionally, the environment can directly signal to the agent

[2]There is also model-based RL, however, the point here is that RL does not always require modeling the environment dynamics like classic AI agents do.

whether the goal has been achieved, which is also a strong assumption that is difficult to satisfy in real-world environments. As a result, these tasks are still often referred to as *closed-world* tasks.

We consider open-world tasks as involving *diverse, rather than fixed, goals* and requiring *semantic-rich actions* in *real-world environments*. In these environments, it is difficult—if possible at all—to formally characterize using symbols whether a goal has been achieved (unlike, for example, mathematical proofs, which can be formalized within the Lean environment [37] that directly verifies whether the proof is valid). Based on our definition, on the one hand, tasks like playing Go & Chess, formal theorem proving, or table manipulation all fall out of the scope of *open-world* tasks as they fail to meet one or more of our requirements. On the other hand, many things may fall into our loose definitions, *e.g.*, robots capable of performing a wide variety of chores in the physical world, agents that can answer various questions over a schema-rich real-world knowledge base (KB), or agents that seek diverse information from the Internet. In all these examples, the agent must handle a range of goals using actions from a highly semantic-rich action space. In addition, these agents operate in environments that may evolve over time and require them to determine when goals are achieved on their own rather than passively receiving signals from the environment. In this thesis, we will mainly focus on tasks within real-world digital environments, including KBs, databases, and websites.

1.2 Motivation

How should we approach the open-world tasks that we discussed above? Due to the massive goal space in open-world tasks, successfully handling them requires the agent to effectively *extrapolate* to novel goals—here we briefly define extrapolation as handling a new goal when the agent already knows how to achieve some seed goals.[3]

[3]It does not really matter how the agent manages to learn the seed goals, be it RL, supervised learning, or even prompting. This is orthogonal to our discussion.

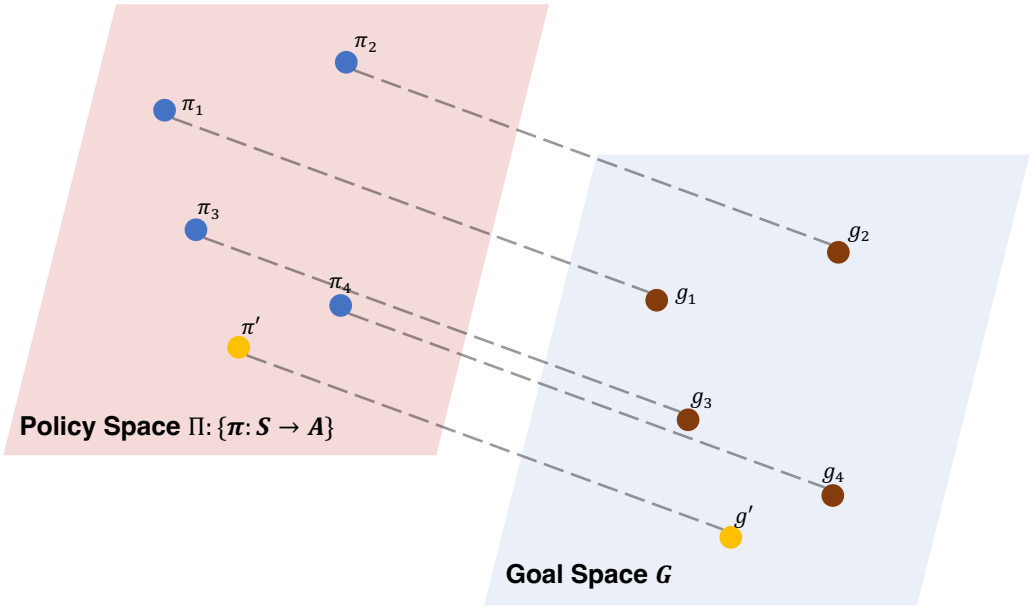

Figure 1.2: An intuitive illustration of our motivation. In a regular RL setting, different policies (*e.g.*, π_1, π_2, ...) are learned in isolation, each optimized according to its own specific reward function. This makes extrapolating to new goals non-trivial. However, if the agent can learn to capture the structure of the goal space and build the mapping function for some seed goal-policy pairs, the agent may potentially extrapolate to a new goal g', given the relationship among g' and the seed goals in the goal space using the same mapping function.

In regular RL settings, the goal is (implicitly) specified through a reward function r. The agent's objective is to learn a policy $\pi \in \Pi \colon \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected cumulative discounted reward defined with r, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space by convention. However, in this standard setting, the policy π is parameterized solely over states $\mathcal{S}$, implying that the learned policy is restricted to a *fixed* goal and cannot generalize across different goals. To address this limitation and develop agents for open-world tasks, goal-conditioned RL (GCRL)—which learns a mapping from $\mathcal{S} \times G$ to $\mathcal{A}$—has emerged as a promising solution [140, 5, 131, 159, 188, 96]. Such goal parameterization results in a single policy that generalizes across the goal space G.

In this thesis, we advocate for explicitly modeling the goal space to achieve better extrapolation, though not necessarily through RL. The intuition of explicit goal modeling is illustrated in Figure 1.2.

By modeling the goal space, we aim to achieve two objectives: 1) ensuring that similar goals share similar policies, and 2) capturing the inherent compositional properties of goals to enable systematic extrapolation. Specifically, there are two core components for achieving this. First, we need to decide on a concrete form of the goal space (*e.g.*, natural language commands, images) along with a representation learning model $\mathcal{M}\colon G \to \mathcal{Z}$ that can preserve the structure of G in a latent space $\mathcal{Z}$. Second, we need a mapping function $f\colon G \to \Pi$ that maps from the goal space G to the policy space Π.[4] [5]

What kind of form the goal space should take? Previous work has tried to design goal specifications with feature vectors [131], images [111], or video demos [188]. However, they are typically restricted to highly constrained environments (*e.g.*, table manipulation tasks with a small set of primitives [188, 131]) due to the limited expressivity of their specifications. To better handle the complexity and diversity in open-world tasks, this thesis argues for the use of natural language for goal specifications. There are two main reasons: 1) first, natural language goal representations $g \in G_{NL}$ naturally provide rich semantics through compositions; 2) modern language models, pre-trained on massive corpora, excel at modeling natural language goals by preserving similarity relationships and compositional structure and can thus serve as an ideal choice for $\mathcal{M}$. By modeling goals in natural language, the agent is expected to extrapolate to novel goals by exploiting the structural and semantic similarities encoded in natural language. For example, the compositional nature of language allows an agent that has learned to perform tasks like *"water plants in kitchen"* and *"fold clothes in bedroom"* to systematically extrapolate to novel combinations such as *"water plants in bedroom"* without a dedicated training process.

[4] $G \to \Pi$ equals $G \to (S \to A)$, which further equals $G \times S \to A$.

[5] Note that, f typically comprises $\mathcal{M}$ as a component (*i.e.*, to first map from G to $\mathcal{Z}$), though in some architectures these components may not be clearly separable.

Regarding the mapping function f, within any single agent implementation, we assume the same function for all goals in this thesis; however, we acknowledge that a mapping function that works well for seed goals may not always extrapolate effectively to new goals. We discuss this in Chapter 7 and leave it to the future work.

1.3 Part I: Understanding Extrapolation Conditioned on Natural Language Goals

To fully understand the potential of goal extrapolation through modeling goals in natural language, we first need a way to systematically evaluate extrapolation. Specifically, it requires to 1) first force some seed goals to the agent, and 2) qualitatively define the relation between new goals and seed goals. In Chapter 2, we discuss our solution to such a systematic evaluation. In particular, we choose knowledge base question answering (KBQA) as our target scenario due to the semantic-rich action space (*i.e.*, relations and types) of modern KBs and the diverse goals (*i.e.*, questions) over the KB. We collect a new benchamark, GRAILQA, with over 60K tasks over the KB and use around 40K of them as the training data so as to force seed goals into the agent. The clear structure of the KB further allows us to clearly define three levels of extrapolation based on the relation between new goals and the goals in our training data: *i.i.d.*, *compositional*, and *zero-shot*. As a result, GRAILQA meet both the two requirements and can serve as a valid testbed to investigate extrapolation with natural language goals. Based on our experiments on GRAILQA, we gain key insights for effective extrapolation. First, pre-trained language models can serve as a much better $\mathcal{M}$, particularly for non-i.i.d. settings, compared with word embeddings like GloVe [130]. Second, regarding the choice of f, unconstrained decoding of actions underperforms constrained decoding, which in turn underperforms discrimination (or ranking).

Built upon the two key insights, we introduce new methods that can better extrapolate agents to new goals on GRAILQA. Specifically, in Chapter 3, we explore the use pre-trained language models

for $\mathcal{M}$ along with a more advanced constrained decoding scheme compared with the limited baseline on GRAILQA. The seamless integration of pre-trained language models and constrained decoding leads to large improvement on GRAILQA. In Chapter 4, we further propose a general framework that can use any type of language models for discrimination, whether encoder-only, decoder-only, or encoder-decoder language models. Also, our method represents the first work that successfully adapts large language models (LLMs) to handle diverse goals over the KB. Even two years later, it remains one of the best performing methods on GRAILQA today.

Our results demonstrate that with modern language models' proficiency in processing natural language, agents can effectively extrapolate to novel goals specified in natural language. This insight points to a promising future: we may be able to optimize agents using a small set of simple tasks and expect them to extrapolate compositional goals, which potentially alleviates the data scarcity issue in training agents for complex tasks.

1.4 Part II: Facilitating Goal-Policy Mapping through Natural Language Prior

A key limitation of the KBQA task discussed in Part I is that the goal description typically operates at a similar level of abstraction as the action space. For example, answering the question *"who painted the painting that James likes?"* requires exactly two actions (*i.e.*, relations): `paints` and `likes`. However, such assumption may not hold in many other open-world tasks. For example, if you instruct a household robot to "`make a pepperoni pizza`," it is not very likely that the robot has an action called `make_pepperoni_pizza`. Instead, it must decompose this high-level goal into numerous primitive actions such as gathering ingredients, preparing the dough, applying sauce, adding toppings, and baking at the appropriate temperature. As a result, there may not always be a direct mapping between the goal space and the policy space as in KBQA. This also means constructing f can be more challenging in open-world tasks without such a strong assumption. In

Part II, we focus on building f with relaxed assumptions, *i.e.*, no direct mapping between goals and actions and less (or *zero*) training data. Particularly, we discuss how to tackle this challenge with the recent advances in LLMs. LLMs encode abundant procedural knowledge about diverse tasks, which can facilitate decision making and thus the construction of f in many scenarios. For example, given the goal "*make a pepperoni pizza*," an LLM can readily decompose it into the necessary subtasks—with natural language descriptions—as we discussed previously.

We explore how to effectively leverage such prior offered by LLMs' language proficiency to effectively build the mapping f from two different angles. In Chapter 5, we first discuss how to effectively apply language-conditioned policy (*i.e.*, CoT reasoning [184]) in the KB and database environment. Our method achieves even better performance than the methods discussed in Part I on a particularly challenging subset of GRAILQA when not using any training data. Additionally, we further show that the encoded natural language prior can also be used for model-based planning. Specifically, we present a new framework that simulates environment dynamics via natural language description in the complex web environment (*i.e.*, given an action and the current observation, output the outcome of the action). The simulated environment enables the agent to effectively search for an optimal sequence of actions within simulation. The proposed framework, WEBDREAMER, achieves considerable improvement compared to the reactive baseline *w/o* search and performs comparably to conducting search directly within a browser with better efficiency and safety.

1.5 Contributions

We briefly summarize the contributions of this thesis as follows:

1. We formulate the problem of extrapolating agents to diverse goals in open-world tasks and highlight two core components: $\mathcal{M}$ and f;

2. We present a benchmark to systematically evaluate extrapolation conditioned on natural language goals, using KBQA as the testbed;

3. We propose new methods that can better extrapolate agents to diverse goals over the KB based on the insights gained from our benchmark, including the first method that successfully adapts LLMs to the KB environment;

4. We show the potential of language-conditioned policy in improving f in complex real-world environments like KB and database using LLMs;

5. We lay out the first model-based planning framework for the real-world web environment with natural language prior encoded in LLMs, which further improves f.

A brief overview of the core design in each chapter is listed in Table 1.1.

	$\mathcal{M}$	f
Chapter 2	word embeddings, encoder-only language models	unconstrained & constrained decoding, discrimination
Chapter 3	encoder-only language models	execution-based constrained decoding
Chapter 4	all types of language models	discrimination
Chapter 5	large language models	language-conditioned policy
Chapter 6	large language models	model-based planning

Table 1.1: A brief overview of the core design in different chapters.

In conclusion, this thesis aims to show the critical role of natural language in developing extrapolative agents. Specifying goals in natural language essentially regularizes the agent's behavior by anchoring it to the concept space derived from language models, which have been pre-trained on

massive corpora. In addition, the natural language prior encoded in LLMs can be effectively used to improve both policy and planning, which can result in a stronger mapping function f. We hope these insights can help agent researchers with a strong footing in natural language processing and LLMs to better understand their role and responsibility in agent research.

Part I: Understanding Extrapolation Conditioned on Natural Language Goals

Chapter 2: Systematically Evaluate Extrapolation Using KBQA

In this chapter, we present a comprehensive testbed that helps us to systematically investigate the extrapolation behavior of modern agents when dealing with diverse goals specified in natural language. Systematic evaluation requires us to first force some seed tasks to the agent and then qualitatively define different types of extrapolation based on the relation between new goals and seed goals. In particular, we curate a large dataset over the knowledge base (KB) environment, leaving out 40K data for training for force the seed goals. Then, we systematically divide extrapolation into three types: *i.i.d.*, *compositional*, and *zero-shot*. This fine-grained categorization allows us to gain deeper insight into extrapolation behaviors in different scenarios. Our findings show that pre-trained language models are particularly helpful for extrapolation under non-i.i.d. settings compared to pre-trained word embeddings. Presumably, language models can do better at extracting the concept space entailed in natural language. In addition, we demonstrate that discrimination serves as the optimal choice for the mapping function f, while unconstrained decoding performs the worst, underperforming both discrimination and constrained decoding.

2.1 Introduction

Question answering on knowledge bases (KBQA) has emerged as a promising technique to provide unified, user-friendly access to knowledge bases (KBs) and shield users from the heterogeneity

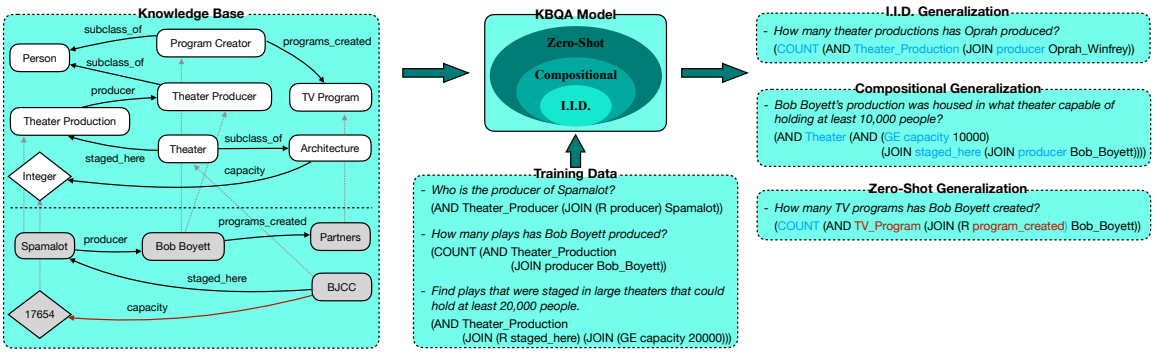

Figure 2.1: On large-scale KBs, collecting sufficient training data for KBQA to ensure i.i.d. distribution at test time is very difficult, if possible at all. We argue that practical KBQA models should have three levels of *built-in generalization* rather than solely relying on training data: (1) *i.i.d. generalization* to questions following the training distribution, (2) *compositional generalization* to novel compositions of schema items seen in training (marked blue), and (3) *zero-shot generalization* to unseen schema items or even domains (marked red). Our definition of generalization is based on the underlying logical forms (shown as S-expressions). Orthogonally, as illustrated by the examples, KBQA models should also have strong generalization to linguistic variation. Figure best viewed in color.

underneath [10, 78, 18, 114]. As the scale and coverage of KBs increase, KBQA is becoming even more important due to the increasing difficulty of writing structured queries like SPARQL.[6]

There has been an array of datasets and models for KBQA in recent years [10, 161, 168, 114, 176, 194, 42, 1, 12]. Most existing studies are (implicitly) focused on the *i.i.d.* setting, that is, assuming training distribution is representative of the true user distribution and questions at test time will be drawn from the same distribution. While it is standard for machine learning, it could be problematic for KBQA on large-scale KBs. First, it is very difficult to collect sufficient training data to cover all the questions users may ask due to the broad coverage and combinatorial explosion. Second, even if one strives to achieve that, *e.g.*, by iteratively annotating all user questions to a deployed KBQA

[6]FREEBASE contains over 100 domains, 45 million entities, and 3 billion facts. GOOGLE KNOWLEDGE GRAPH has amassed over 500 billion facts about 5 billion entities [163].

system, it may hurt user experience because the system would keep failing on new out-of-distribution questions not covered by existing training data in each iteration.

Therefore, we argue that *practical KBQA models should be built with strong generalizability to out-of-distribution questions at test time*. More specifically, we propose three levels of generalization: *i.i.d.*, *compositional*, and *zero-shot* (Figure 2.1). In addition to the standard i.i.d. generalization, KBQA models should also generalize to novel compositions of seen schema items (relations, classes, functions). For example, if a model has been trained with questions about relations like `producer`, `staged_here`, `capacity`, classes like `Theater`, and functions like `GE`, it should be able to answer complex questions involving all these schema items even though this specific composition is not covered in training. Furthermore, a KBQA model may also encounter questions about schema items or even entire domains that are not covered in training at all (*e.g.*, `TV_Program` and `program_created`) and needs to generalize in a zero-shot fashion [124].

High-quality datasets are important for the community to advance towards KBQA models with stronger generalization. In addition to providing a benchmark for all three levels of generalization, an ideal dataset should also be large-scale, diverse, and capture other practical challenges for KBQA such as entity linking, complex questions, and language variation. However, existing KBQA datasets are usually constrained in one or more dimensions. Most of them are primarily focused on the i.i.d. setting [10, 195, 16, 168]. GRAPHQ [161] and QALD [114] could be used to test compositional generalization but not zero-shot generalization. They are also relatively small in scale and have limited coverage of the KB ontology. SIMPLEQ [16] is large but only contains single-relational questions with limited diversity. A quantitative comparison is shown in Table 2.1.

Therefore, we construct a new large-scale, high-quality dataset for KBQA, GRAILQA (Strongly Generalizable Question Answering), that supports evaluation of all three levels of generalization. It contains 64,331 crowdsourced questions involving up to 4 relations and functions like counting,

comparatives, and superlatives, making it the largest KBQA dataset with complex questions to date. The dataset covers all the 86 domains in FREEBASE COMMONS, the part of FREEBASE considered to be high-quality and ready for public use, and most of the classes and relations in those domains. Furthermore, the questions in our dataset involve entities spanning the full spectrum of popularity from `United_States_of_America` to, *e.g.*, `Tune_Hotels` — a small hotel chain mainly operated in Malaysia. To make the dataset even more diverse and realistic, we collect common surface forms of entities, *e.g.*, *"Obama"* and *"President Obama"* for `Barack_Obama`, via large-scale web mining and crowdsourcing and use them in the questions. Finally, we develop multiple quality control mechanisms for crowdsourcing to ensure dataset quality, and the final dataset is contributed by 6,685 crowd workers with highly diverse demographics.

In addition, we also study important factors towards stronger generalization and propose a novel KBQA model based on pre-trained language models like BERT [40]. We first show that our model performs competitively with existing models: On GRAPHQ, our model sets a new state of the art, beating prior models by a good margin (3.5%). On GRAILQA, our model significantly outperforms a state-of-the-art KBQA model. More importantly, the combination of our dataset and model enables us to thoroughly examine several challenges in KBQA such as search space pruning and language-ontology alignment. The comparison of different variants of our model clearly demonstrates the critical role of BERT in compositional and zero-shot generalization. To the best of our knowledge, *this work is among the first to demonstrate the key role of pre-trained contextual embeddings such as BERT at multiple levels of generalization for KBQA*. Finally, we also show that our dataset could serve as a valuable pre-training corpus for KBQA in general: pre-trained on our dataset, our model can generalize to other datasets (WEBQ) and reach similar performance fine-tuned with only 10% of the data, and can even generalize in a zero-shot fashion across datasets. To summarize, the key contributions of this chapter are three-fold:

- We present the first systematic study on three levels of generalization, *i.e.*, *i.i.d.*, *compositional*, and *zero-shot*, for KBQA and discuss their importance for practical KBQA systems.
- We construct and release a large-scale, high-quality KBQA dataset containing 64K questions with diverse characteristics to support the development and evaluation of KBQA models with stronger generalization at all three levels.
- We propose a novel BERT-based KBQA model with competitive performance and thoroughly examine and demonstrate the effectiveness of pre-trained contextual embeddings in generalization. We also present fine-grained analyses which point out promising venues for further improvement.

2.2 Background

2.2.1 Knowledge Base

A knowledge base consists of two parts, an ontology $\mathcal{O} \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$ and the relational facts $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times (\mathcal{C} \cup \mathcal{E} \cup \mathcal{L})$, where $\mathcal{C}$ is a set of classes, $\mathcal{E}$ is a set of entities, $\mathcal{L}$ is a set of literals and $\mathcal{R}$ is a set of binary relations. An example is shown in Figure 2.2, where the top part is a snippet of the FREEBASE ontology and the bottom part is some of the facts. We base our dataset on the latest version of FREEBASE. Although FREEBASE has stopped getting updated, it is still one of the largest publicly available KBs, and its high quality from human curation makes it a solid choice for benchmarking KBQA. We use its COMMONS subset which contains 86 domains, 2,038 classes, 6,265 relations, and over 45 million entities.

2.2.2 Three Levels of Generalization: Definition

The definitions are based on the underlying logical form of natural language questions. We denote $\mathcal{S}$ as the full set of *schema items* that includes $\mathcal{R}$, $\mathcal{C}$, and optionally a set of language-specific constructs from the meaning representation language (*e.g.*, SPARQL). In our case we include the set

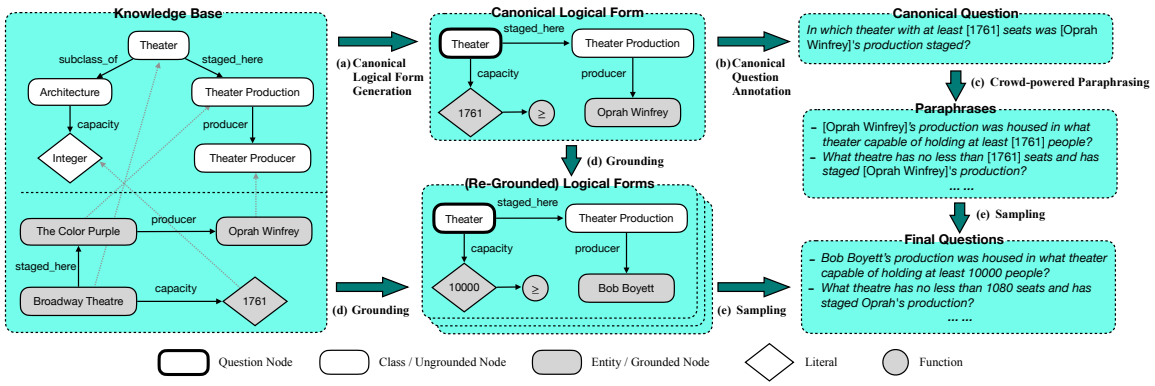

Figure 2.2: Our data collection pipeline with illustration of how one example question in Figure 2.1 is derived: (a) Generate canonical logical forms from the given KB up to the specified complexity. (b) Experts annotate canonical questions for the canonical logical forms. Entities and literals are enclosed in brackets so that they can be easily replaced. (c) Get high-quality and diverse paraphrases via crowdsourcing. (d) Generate different logical forms from each canonical logical form with different entity groundings. (e) Sample different combinations of logical form and paraphrase to generate the final questions. We additionally mine common surface forms of entities from the web, *e.g.*, *"Oprah"* for `Oprah_Winfrey`, and use them in the final questions to make entity linking more realistic.

of functions (Section 2.3). Note that entities and literals are not included. Denote $\mathcal{S}_{train}$ as the set of schema items in any of the training examples and $\mathcal{S}_q$ that of a question q. A qualifying test set $\mathcal{Q}$ for each level of generalization is defined as:

- **I.I.D. generalization**: $\forall q \in \mathcal{Q}, \mathcal{S}_q \subset \mathcal{S}_{train}$. In addition, the test questions follow the training distribution, *e.g.*, randomly sampled from the training data.
- **Compositional generalization**: $\forall q \in \mathcal{Q}, \mathcal{S}_q \subset \mathcal{S}_{train}$, however, the logical form of q is not covered in training.
- **Zero-shot generalization**: $\forall q \in \mathcal{Q}, \exists s \in \mathcal{S}_q, s \in \mathcal{S} \setminus \mathcal{S}_{train}$.

The levels of generalization represent the expectation on KBQA models: A model should minimally be able to handle questions i.i.d. to what it has been trained with. One step further,

it should also generalize to novel compositions of the seen constructs. Ideally, a model should also handle novel schema items or even entire domains not covered by the limited training data, which also includes compositions of novel constructs. By explicitly laying out the different levels of generalization, we hope to encourage the development of models built with stronger generalization. Orthogonal to these, practical KBQA models should also have strong generalization to language variation, *i.e.*, different paraphrases corresponding to the same logical form. The dataset we will introduce next also puts special emphasis on this dimension.

2.3 Data

2.3.1 Data Collection

	Questions	Canonical LF	Domains	Relations	Classes	Entities	Literals	Generalization Assumption
WEBQ[10, 195]	4,737	794	56	661	408	2,593	47	i.i.d.
COMPLEXWEBQ[168]	34,689	6,236	61	1,181	501	11,840	996	i.i.d.
SIMPLEQ[16]	108,442	2,004	76	2,004	741	89,312	0	i.i.d.
QALD[114]	558	558	N/A	254	119	439	28	comp.
GRAPHQ[161]	5,166	500	70	596	506	376	81	comp.
GRAILQA	64,331	4,969	86	3,720	1,534	32,585	3,239	i.i.d.+ comp.+ zero-shot

Table 2.1: Comparison of KBQA datasets. The number of distinct canonical logical forms (LFs) provides a view into the diversity of logical structures. For example, even though SIMPLEQ appears to be the largest, its diversity is limited because it only contains single-relational logical forms with no function (*e.g.*, over 3,000 questions asking about the `place_of_birth` of different persons). Datasets other than GRAILQA are also likely to have test samples including schema items not covered in training, but that is not sufficient to support systematic evaluation of zero-shot generalization.

[182] propose the OVERNIGHT approach which collects question-logical form pairs with three steps: (1) generate logical forms from a KB, (2) convert logical forms into canonical questions, and (3) paraphrase canonical questions into more natural forms via crowdsourcing. [161] extend the OVERNIGHT approach to large KBs like FREEBASE with algorithms to generate logical forms that correspond to meaningful questions from the massive space. However, the data collection in

Question	Domain	Answer	# of Relations	Function
Beats of Rage is a series of games made for which platform?	Computer Video Game	DOS	1	none
Which tropical cyclone has affected Palau and part of Hong Kong?	Location, Meteorology	Typhoon Sanba	3	none
Marc Bulger had the most yards rushing in what season?	Sports, American Football	2008 NFL Season	3	superlative
How many titles from Netflix have the same genre as The Big Hustle?	Media Common	20,104	2	count
What bipropellant rocket engine has less than 3 chambers?	Spaceflight	RD-114 RD-112, ...	1	comparative

Table 2.2: Example questions from GRAILQA.

[161] was entirely done by a handful of expert annotators, which significantly limits its scalability and question diversity. We build on the approach in [161] and develop a crowdsourcing framework with carefully-designed quality control mechanisms, which is much more scalable and yield more diversified questions. As a result, we are able to construct a dataset one order of magnitude larger with much better coverage of the FREEBASE ontology (Table 2.1). Our data collection pipeline is illustrated in Figure 2.2.

Canonical logical form generation. We leverage the logical form generation algorithm from [161], which randomly generates diverse logical forms. The algorithm guarantees some important properties of the generated logical forms such as well-formedness and non-redundancy. It first traverses the KB ontology to generate graph-shaped templates that only consist of classes, relations, and functions, and then ground certain nodes to compatible entities to generate logical forms in their meaning representation called *graph query*[7]. At this stage, we only ground each template with one set of compatible entities to generate one *canonical logical form* and use it to drive the subsequent steps. Following prior work, we generate logical forms containing up to 4 relations and optionally containing one function selected from counting, superlatives (`argmax`, `argmin`), and comparatives ($>$, $\geq$, $<$, $\leq$). The exemplar logical form in Figure 2.2 has 3 relations and one function.

[7]Refer to [161] for the syntax and semantics of graph query.

Each canonical logical form is validated by a graduate student against the criterion – *could we reasonably expect a real human user to ask this question?* We only keep the valid ones, which reduces the number of artificial questions that are common in other KBQA datasets with generated logical forms such as COMPLEXWEBQ [168] and SIMPLEQ [16] and makes the dataset more realistic. The overall approval rate at this step is 86.5%.

Canonical question annotation. Each validated canonical logical form is annotated with a canonical question by a graduate student, which is then cross-validated by another student to ensure its fidelity and fluency. The graduate students are trained with detailed materials about the annotation task. To further facilitate the task, we develop a graphical interface which displays an interactive graphical visualization of the logical form. One can click on each component to see auxiliary information such as a short description and its domain/class. Annotators are required to enclose entities and literals in brackets (Figure 2.2).

Crowd-powered paraphrasing. We use Amazon Mechanical Turk to crowdsource paraphrases of the canonical questions and limit our task to native English speakers with at least 95% task approval rate. We develop a framework with automated quality control mechanisms that contains three tasks:

1. *Paraphrasing*. A crowd worker is presented with a canonical question as well as auxiliary information such as topic entity description and answer to the question, and is tasked to come up with a plausible and natural paraphrase. Entities and literals enclosed with brackets are retained verbatim in the paraphrase. The system keeps running until 5 *valid* paraphrases (Task 2) are obtained for each canonical question. We pay $0.10 per paraphrase.

2. *Cross-validation*. Each paraphrase is then judged by multiple independent workers (the original author is excluded) on its fluency and fidelity (*i.e.*, semantic equivalence) to the corresponding canonical question. Each paraphrase gets 4 judgements on average and those with fidelity

approval rate below 75% *or* fluency approval rate below 60% are discarded. Overall 17.4% of paraphrases are discarded. A worker judges 10 paraphrases a time, one of which is a *control question*. We manually craft an initial set of paraphrase-canonical question pairs known to be good/bad as control questions and gradually expand the set with new pairs validated by crowd workers. We ask workers to reconsider their validation decision of the whole batch when they have failed the latent control question, and we found that by doing so spammers or low-quality workers tend to quit the task and we end up with mostly high-quality workers. We pay \$0.15 per batch.

3. *Entity surface form mining*. We collect a list of common surface forms ranked by frequency for each entity from FACC1 [47], which identifies around 10 billion mentions of FREEBASE entities in over 1 billion web documents from ClueWeb. For example, some common surface forms (and frequency) for `Barack_Obama` are *"obama"* (21M), *"barack obama"* (5.3M), and *"barack hussein obama"* (101K). We then ask at least three crowd workers to select true surface forms from the mined list, and those selected by less than 60% of the workers are discarded. We pay \$0.05 per task.

Grounding and sampling. More logical forms are generated by grounding each canonical logical form with *compatible* entity groundings.[8] We do controlled sampling to generate the final questions: From the pool of logical forms and paraphrases associated with the same canonical logical form, we sample one from each pool at a time to generate a question (Figure 2.2). We start with uniform weights and each time a logical form or paraphrase is selected, its weight is divided by ρ_l and ρ_p, respectively. We set ρ_l to 2 and ρ_p to 10 to enforce more linguistic diversity. Finally, we randomly

[8]An entity grounding, *e.g.*, (`Bob Boyett`, `10000`), is compatible with a canonical logical form if the re-grounded logical form yields a non-empty denotation.

replace entity surface forms with the ones mined in Task 3 (if there is any). This way, we are able to generate a large-scale dataset with carefully-controlled quality and diversity.

2.3.2 Dataset Analysis

In total we have collected 4,969 canonical logical forms and 29,457 paraphrases (including the canonical questions). The final dataset contains 64,331 question-logical form pairs after sampling, which is by far the largest dataset on KBQA with complex questions. We note that such sampling is a unique feature of GRAILQA because of our data collection design. A detailed comparison with existing datasets is shown in Table 2.1 and some example questions are shown in Table 2.2. GRAILQA has significantly broader coverage and more unique canonical logical forms than existing dataset except COMPLEXWEBQ.[9] GRAILQA is also the only KBQA dataset that explicitly evaluates zero-shot generalization (Section 2.5.1). Question distributions over different characteristics are shown below:

# of Relations				**Function**				**Answer Cardinality**	
1	2	3	4	none	count	super.	comp.	1	> 1
44,340	16,610	3,254	127	53,526	3,463	4,111	3,231	44,044	20,287

Quality. As a qualitative study on data quality, we manually analyze 100 randomly sampled canonical logical forms, their canonical questions, and the associated paraphrases: 3 of the 100 canonical questions have mismatching meaning with the canonical logical form, mostly due to misinterpreting the directionality of some relation, and 12 of the 576 paraphrases do not match the corresponding canonical question, leading to 3% error rate of canonical question annotation, 2.1% error rate of paraphrasing, and 5.6% error rate overall. All the examined paraphrases are reasonably fluent.

[9]Logical forms in COMPLEXWEBQ are automatically extended from WEBQ with extra SPARQL statements. This adds structural diversity but may lead to unnatural questions.

Linguistic diversity. The dataset is contributed by 11 graduate students and 6,685 crowd workers with diverse demographics in terms of age group, education background, and gender. A common concern with crowd-powered paraphrasing as a means of data collection is lack of linguistic diversity — crowd workers may be biased towards the canonical question [60]. However, that may not be the case for GRAILQA. The average Jaccard similarity between each paraphrase and the corresponding canonical question, when all lower-cased, is 0.569 and 0.286 for unigrams and bigrams, respectively. The average Levenshtein edit distance is 26.1 (with average total length of 50.1 characters). The same statistics between the paraphrases of the same canonical question are 0.527, 0.257, and 28.2. The dissimilarity is even higher when excluding entities. We believe this is a decent level of linguistic diversity and we attribute it to our diverse crowd workers and quality control mechanisms.

Entity linking. GRAILQA also features more realistic and challenging entity linking thanks to our large-scale mining of entity surface forms. Common types of surface forms include acronym (*"FDA"* for `Food_and_Drug_Administration`), commonsense (*"Her Majesty the Queen"* for `Elizabeth_II`), and colloquial vs. formal (*"Obama vs. Romney"* for `United_States_Presidential_Election_2012`). In general the mined surface forms are the more typical, colloquial way of referring to an entity than its formal name in the KB, which makes the questions more realistic. We note that this is an important challenge towards practical KBQA systems but is largely neglected in existing KBQA datasets. For example, [192] shows that, for the WEBQ dataset, simple fuzzy string matching is sufficient for named entity recognition (NER) due to the way the dataset is constructed.

2.3.3 Logical Form in S-expression

In addition to graph query, we provide an alternative linearized version of it in S-expressions, which is needed to apply the mainstream sequence-to-sequence (Seq2Seq) neural models [68, 42, 203]. We find that S-expression provides a good trade-off on compactness, compositionality, and

readability, but one may choose any other formalism for linearization such as λ-DCS [92] or directly use the corresponding SPARQL queries. Every graph query can be easily converted to an equivalent S-expression. For example, The graph query in Figure 2.2(d) can be converted into `(AND Theater (AND (GE capacity 10000) (JOIN staged_here (JOIN producer Bob_Boyett))))`. Both graph queries and S-expressions can be easily converted into SPARQL queries to get answers. We will use S-expressions as logical form in our modeling.

2.4 Modeling

In this section, we discuss some of the challenges arising from non-i.i.d. generalization and potential solutions. The proposed models, combined with GRAILQA, will enable us to evaluate and compare different strategies for stronger generalization.

Compositional and zero-shot generalization pose two unique challenges compared with i.i.d. generalization: *large search space* and *language-ontology alignment*. The first challenge is the significantly larger search space. Under the i.i.d. assumption, a model only needs to consider the part of the ontology observed during training. For zero-shot generalization, however, one could not assume that to hold at test time and needs to consider the entire ontology. Models that build their vocabulary solely from training data (*e.g.*, [203]) would almost certainly fail at zero-shot generalization. *Question-specific search space pruning* is thus more important than it is in the i.i.d. setting. Secondly, a major challenge in KBQA is to build a precise alignment between natural language and schema items in the ontology in order to understand what each question is asking about. While it is already important for i.i.d. generalization, it becomes even more critical for non-i.i.d. generalization: For zero-shot generalization, one needs to understand the unseen schema items in order for it to possibly handle the corresponding questions. Even for compositional generalization, where the schema items are covered in training, precise language-ontology alignment is still critical

in order for the model to generate novel compositions other than something it has memorized in training. Conventional methods [10, 1, 18] use web mining to build a lexicon from natural language phrases to KB schema items, which may be biased towards popular schema items. The emergence of pre-trained contextual embeddings such as BERT [40] provides an alternative solution to building language-ontology alignment. Next, we propose a BERT-based KBQA model which will enable us to examine the effectiveness of pre-trained contextual embeddings in non-i.i.d. generalization.

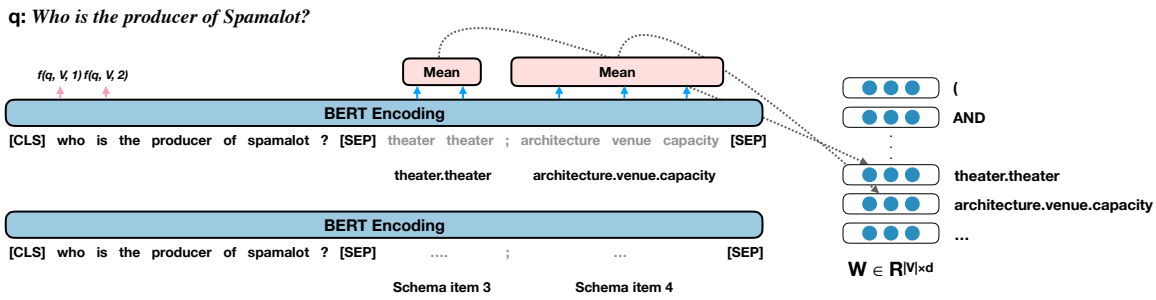

Figure 2.3: An overview of using BERT to jointly encode the input question and schema items in the decoder vocabulary. We evenly split all items in the vocabulary into chunks and concatenate each chunk with the question one by one. For the purpose of visualization, here we set the chunk size as 2. The first chunk contains two schema items, `theater.theater` and `architecture.venue.capacity`. For each item in $\mathcal{V}$, we compute its embedding in $\mathbf{W}$ by averaging the last layer's output of BERT of all its word-pieces. To compute $f(q, \mathcal{V}, t)$, we simply take the last layer's output of BERT in the first chunk for the t-th word-piece of the question. Note that, we show whole words instead of actual word-pieces in the figure for brevity.

2.4.1 Model Overview

Our goal is to learn a model that maps an input question $q = x_1, ..., x_{|q|}$ to its corresponding logical form $a = y_1, ..., y_{|a|}$. The entire model is based on Seq2Seq [165, 8], which is commonly used in many semantic parsing tasks [42, 68, 66, 52, 204]. Seq2Seq comprises an encoder that encodes input q into vectors and a decoder that autoregressively output one token $y' \in \mathcal{V}$ conditioned on the

input representation, where $\mathcal{V}$ is the decoding vocabulary. Specifically, the conditional probability $p(a|q)$ is decomposed as:

$$p(a|q) = \prod_{t=1}^{|a|} p(y_t|y_{<t}, q), \tag{2.1}$$

where $y_{<t} = y_1, ..., y_{t-1}$. Our encoder and decoder are two different recurrent neural networks with long short-term memory units (LSTM) [61]. LSTM recursively processes one token at each time step as the following function:

$$\mathbf{h}_t = LSTM(\mathbf{h}_{t-1}, \mathbf{g}_t), \tag{2.2}$$

where $\mathbf{g}_t$ is the representation of the input token at the t-th step, and $\mathbf{h}_{t-1}$ is the hidden state from last time step. During encoding, $\mathbf{g}_t = f(q, \mathcal{V}, t)$, with f being a function takes $q, \mathcal{V}, t$ as input and returns the embedding for x_t; during decoding, $\mathbf{g}_t = [\mathbf{W}]_{y_t}$, which denotes the the embedding corresponding to y_t in the embedding matrix $\mathbf{W} \in \mathbf{R}^{|\mathcal{V}| \times d}$. We will elaborate how we use BERT to define f and $\mathbf{W}$ in the next subsection.

During decoding, the model computes the probability for each output token based on the current hidden state $\mathbf{h}_t$ and $\mathbf{W}$:

$$p(y_t|y_{<t}, q) = [Softmax(\mathbf{W}\mathbf{h}_t)]_{y_t}. \tag{2.3}$$

This means we tie the input and output embeddings using $\mathbf{W}$ in Seq2Seq terminology. In this way, we can assign semantic meanings to words from pre-trained embeddings, which can facilitate the open vocabulary learning [177].

2.4.2 BERT Encoding

Now we discuss how we use BERT to compute $\mathbf{W}$ and $f(q, \mathcal{V}, t)$. BERT has demonstrated its effectiveness in learning good alignment in cross-domain text-to-SQL parsing with its contextual representation. Specifically, in text-to-SQL, Hwang et al.[66] successfully apply BERT by concatenating the question and all vocabulary items together to construct the input for BERT. However, the vocabulary in KBQA is much larger. Directly doing this will exceed BERT's maximum input length of 512 tokens. To address this, we split items from $\mathcal{V}$ into chunks and then concatenate each chunk independently with q, and feed them to BERT one by one. Figure 2.3 depicts an example where each chunk has 2 items from $\mathcal{V}$. Specifically, q is separated with each chunk by a special token "[SEP]", and the items inside the same chunk are delimited by ";". Then $f(q, \mathcal{V}, t)$ can be simply defined as the t-th output from BERT for the first chunk. For $\mathbf{W}$, each row in it is computed by taking average over BERT's output for the constituting words of the corresponding item. For example, the row corresponding to `architecture.venue.capacity` in $\mathbf{W}$ is computed by averaging the output from BERT for word *"architecture"*, *"venue"* and *"capacity"* in the first chunk as shown in Figure 2.3.

Vocabulary construction. We have not talked about what constitutes $\mathcal{V}$. The most trivial way is just to include all schema items from KB ontology to deal with zero-shot generalization. However, this will not only lead to an enormous search space during decoding that makes the prediction very difficult, but also makes it extremely time-consuming for training as we need to create a large number of chunks to feed to BERT for every single question. To reduce the size of $\mathcal{V}$, we leverage entities identified from q as anchors in the KB, and choose to only include KB items that are reachable by at least one of the anchor entities within 2 hops in KB. We refer to this as vocabulary pruning (VP), which is applied during both training and inference. This means we have a dynamic $\mathcal{V}$ for different input q. Note that all the identified entities, functions, and syntax constants used in S-expressions are

also included. We similarly get the representation for an entity based on the output from BERT that corresponds to its surface form.

2.4.3 Entity Linking

Entity linking in most of the existing KBQA datasets is not a major challenge, and exhaustive fuzzy string matching [192] may suffice to achieve a reasonable performance. However, the entities in GRAILQA span the full spectrum of popularity and appear in surface forms mined from the web, which is more realistic but also makes entity linking more challenging. We use a BERT-based NER system[10] and train it on our training set. For entity disambiguation from the identified mentions, we simply select the most popular one(s) based on FACC1. As a comparison, we also try Aqqu [9], a rule-based entity linker using linguistic and entity popularity features and achieves high accuracy on WEBQ. The entity linking results on GRAILQA is shown as follows: The BERT-based entity linker

	Recall	**Precision**	**F1**
Aqqu	**77.8**	9.7	17.2
BERT	77.0	**68.0**	**72.2**

is slightly worse in recall but much better in precision. We will therefore use the BERT-based entity linker afterwards.

2.4.4 Inference

We propose two different modes of doing inference using our model, namely TRANSDUCTION and RANKING. For TRANSDUCTION, we simply use the model in the normal way, *i.e.*, use the model to autoregressively predict one token from $\mathcal{V}$ at each time step until the model predicts the stop token. For RANKING, instead of using Seq2Seq as a generator we use it as a ranker to score

[10]https://github.com/kamalkraj/BERT-NER

each candidate logical form and return the top-ranked candidate. We employ a simple yet effective strategy to generate candidate logical forms: We enumerate all logical forms, optionally with a `count` function, within 2 hops starting from each entity identified in the question.[11] The recall is 80% on GRAILQA. Questions with superlatives and comparatives often do not have a topic entity and are therefore not covered. RANKING can prune the search space more effectively than the first one, while TRANSDUCTION is more flexible and can handle more types of questions.

Effectiveness on existing dataset. Before getting to experiments on our GRAILQA dataset, we conduct an experiment on the existing dataset GRAPHQ to show the competitive performance of our model. Our RANKING model achieves an F1 of 25.0%, which significantly outperforms the prior art SPARQA [164] by 3.5 percent. With this superior performance, we believe our new model is reasonable to serve as a strong baseline on GRAILQA and support the subsequent investigations on three levels of generalization.

2.5 Experiments

	Overall		I.I.D.		Compositional		Zero-shot	
	EM	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
QGG [81]	–	36.7	–	40.5	–	33.0	–	36.6
TRANSDUCTION	33.3	36.8	51.8	53.9	31.0	36.0	25.7	29.3
– VP	26.6	29.6	40.1	42.7	25.6	30.0	20.6	23.4
– BERT	17.6	18.4	50.5	51.6	16.4	18.5	3.0	3.1
– VP – BERT	15.4	16.1	48.3	49.1	13.5	15.3	1.0	1.3
RANKING	**50.6**	**58.0**	59.9	67.0	**45.5**	**53.9**	**48.6**	**55.7**
– BERT	39.5	45.1	**62.2**	**67.3**	40.0	47.8	28.9	33.8

Table 2.3: Overall results. "– VP" denotes without vocabulary pruning. "– BERT" denotes using GloVe instead of BERT.

[11] There are prohibitively many candidates with 3 hops.

2.5.1 Experimental Setup

Data split. We split GRAILQA to set up evaluation for all three levels of generalization. Specifically, our training/validation/test sets contain about 70%/10%/20% of the data, which correspond to 44,337, 6,763 and 13,231 questions, respectively. For the validation and test sets, 50% of the questions are from held-out domains not covered in training (**zero-shot**), 25% of the questions correspond to canonical logical forms not covered in training (**compositional**), and the rest 25% are randomly sampled from training (**i.i.d.**). The i.i.d. and compositional subsets have the additional constraint that the involved schema items are all covered in training. For the zero-shot subset, 5 domains are held out for validation and 10 for test.

Evaluation metrics. We use two standard evaluation metrics. The first one is *exact match accuracy* (EM), *i.e.*, the percentage of questions where the predicted and gold logical forms are semantically equivalent. To determine semantic equivalence, we first convert S-expressions back to graph queries and then determine graph isomorphism.[12] Unlike string-based or set-based EM [200], our graph isomorphism based EM is both sound and complete. In addition, we also report the F1 score based on the predicted and gold answer sets [10]. This is better suited for models that directly predict the final answers without any intermediate meaning representation and gives partial credits to imperfect answers. We host a local SPARQL endpoint via Virtuoso to compute answers.

2.5.2 Models

Our primary goal of the experiments is to thoroughly examine the challenges of different levels of generalization and explore potential solutions. Therefore, we will evaluate an array of variants of our models with different strategies for search space pruning and language-ontology alignment. To better situate our models in the literature and demonstrate their competitive performance, we

[12] `https://networkx.github.io`

also adapt QGG [81], the state-of-the-art model on COMPLEXWEBQ and WEBQ, to GRAILQA. We have also looked into models developed based on QALD or LC-QUAD but the adaptation cost would be too high because most of the source codes are not available. All models use the same entity linker (Section 2.4.3).

Our model variants. For both TRANSDUCTION and RANKING, we introduce a variant that uses GloVe embeddings[13] instead of BERT for language-ontology alignment. Specifically, we use the average GloVe embedding for each schema item instead of the encodings from BERT. Unlike BERT, GloVe embeddings are not contextual and are thus not jointly encoded with the current natural language question. This variant will then allow us to examine the role of contextual embedding in generalization. For TRANSDUCTION, we additionally introduce a variant that does not employ entity-based vocabulary pruning (VP). This is not applicable to RANKING because RANKING always relies on the identified entities for candidate generation. We use uncased BERT-base and fine-tune BERT but fix GloVe (similar to previous work [52]), because we find fine-tuning GloVe makes it slightly worse.

QGG. This model learns to generate query graphs from question-answer pairs using reinforcement learning. BERT is also used but only to get a matching score between question and logical form via joint encoding, which is served as one of the seven hand-crafted features used for ranking. To train this model on GRAILQA, we use the same configuration from the original paper for WEBQ, *i.e.*, considering up to 2-hop relations in the KB for candidate generation. This can cover 80% of the training questions in GRAILQA. Increasing it to 3 hops will take a few months (estimated) to train this model due to the large number of entities in GRAILQA. We only report F1 for QGG because it uses a different meaning representation.

[13]We choose to use GloVe-6B-300d.

2.5.3 Results

Overall Evaluation

We show the overall results in Table 2.3. RANKING achieves the best overall performance on GRAILQA. Both of our models outperform QGG, demonstrating their competitive performance. We also observe a significant performance drop on all the variants of our models, which suggest that both BERT encoding and VP play an important role.

Next we drill down to different levels of generalization. The results clearly demonstrate the key role of contextualized encoding via BERT for compositional and zero-shot generalization: While BERT and GloVe perform similarly in i.i.d. generalization, using GloVe instead of BERT, TRANSDUCTION's F1 drops by 17.5% in compositional generalization and by 26.2% in zero-shot generalization. For RANKING, it also brings a 21.9% drop in zero-shot generalization. We do a more in-depth analysis on TRANSDUCTION and confirm that this is mainly because BERT enables better generalization to unseen schema items. For example, for question *"What home games did the Oakland Raiders play?"* about an unseen domain `american_football`, TRANSDUCTION can find the correct relation `american_football.football_game.home_team`, while TRANSDUCTION without BERT is confused by the wrong relation `sports.sports_team.arena_stadium`, which is seen during training. This is a common type of error by GloVe-based models but much less frequent by BERT-based models, which shows how pre-trained contextual embeddings help address the *language-ontology alignment* challenge. Since QGG is also a ranking model that uses BERT, it is not surprising it also performs reasonably in compositional and zero-shot generalization.

On the other hand, RANKING outperforms TRANSDUCTION significantly in compositional and zero-shot generalization. This is largely due to RANKING's effectiveness in *search space pruning*, *i.e.*, RANKING prunes the search space based on the each identified entity's neighboring facts, while TRANSDUCTION only uses the ontology, which is less discriminating. VP helps TRANSDUCTION

to some extent, but still not quite up to the same level of effective pruning RANKING enjoys. This also provides a plausible explanation for the interesting observation that BERT seems to play a less significant role in compositional generalization in ranking models than in transductive models — because of the effective search space pruning, RANKING's candidate set includes much less of the logical forms seen in training that may be confusing to the model.

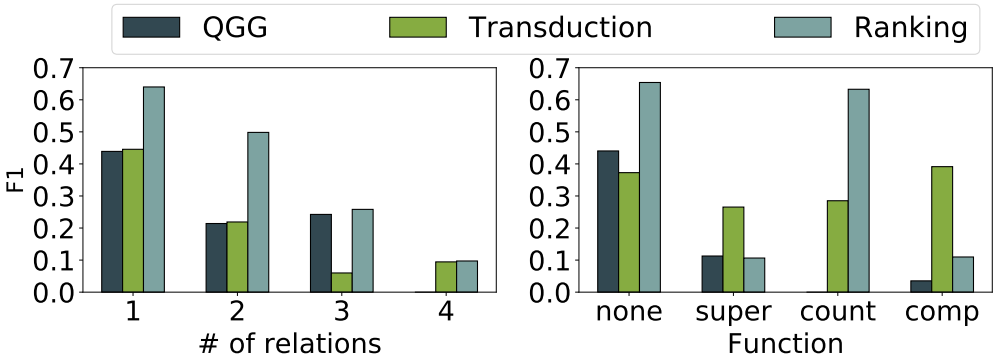

Figure 2.4: Fine-grained results.

Fine-Grained Evaluation

We now present a more fine-grained analysis along several other dimensions in Figure 2.4.

Structural complexity. The performance of all models degrades rapidly as questions become more complex. Note that, though our RANKING model only generates candidates with up to 2 relations, it still has a chance to get partially correct on more complex questions. In fact, it even outperforms the more flexible TRANSDUCTION model on questions with 3 relations, which again demonstrates the importance of effective search space pruning.

Function. Ranking models (including RANKING and QGG) rely on topic entities for candidate generation which are often absent in questions with comparatives or superlatives, *e.g.*, *"which chemical*

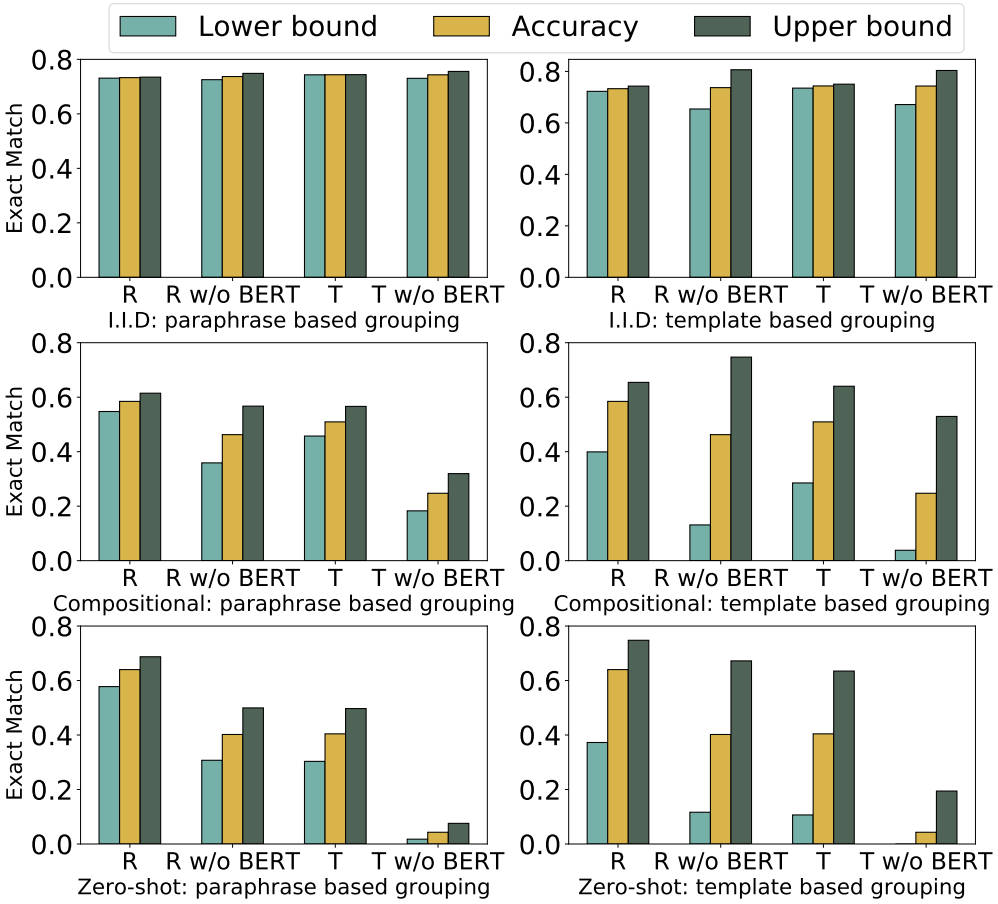

Figure 2.5: Robustness analysis. T denotes TRANSDUCTION and R denotes RANKING. For robustness analysis, we assume correct entities are given.

element was first discovered?" Their performance suffers as a result. QGG gets zero F1 on questions that requires counting since it only returns entities as answer. On the other hand, TRANSDUCTION performs significantly better than the other two models on superlatives and comparatives due to its flexibility in generating all types of logical forms, though that comes with a penalty on simpler questions. Better guided search for both RANKING and TRANSDUCTION may be a promising future direction to enjoy the best of both worlds.

Robustness Analysis

A good KBQA model should be robust to different paraphrases and entity groundings (*e.g.*, should not succeed at *"Where is the Trump tower?"* but fail at *"Where is the Tune Hotels?"*). GRAILQA provides an opportunity to directly test that. We create different groupings of questions: *grouping by paraphrase* — questions that are the same except for their entity groundings fall into the same group, and *grouping by template* — questions from the same canonical logical form but with different groundings or paraphrases are grouped together. Each grouping provides upper and lower bounds for evaluating model robustness. The upper bound is derived by treating all the questions in a group as correct if any of them is correctly answered, while the lower bound is derived by treating all the questions as incorrect if any of them is incorrectly answered. The true accuracy should lie between the bounds. The closer the three are, the more robust a model is to the corresponding mutation. To reduce the number of variables in this analysis, we assume perfect entity linking and eliminate entity linking errors. The results are shown in Figure 2.5, which clearly show that BERT-based models are generally more robust than GloVe-based models (reflected by the smaller gap between their upper and lower bounds) and provides further explanation to its superior performance in non-i.i.d. generalization. It is interesting to observe that the RANKING model yields a higher upper bound with GloVe than with BERT in non-i.i.d. generalization, which shows that the major problem with GloVe is its robustness to variations.

2.5.4 Error Analysis

We analyze 100 randomly sampled questions that our RANKING model gets wrong to discuss venues for future improvement. We summarize the errors into the following categories.

Coverage limitation (34%): Due to the diversity of questions in GRAILQA in terms of both complexity and functions, candidates generated by RANKING can only cover around 80% of the questions,

as it can only enumerate up to 2-relational logical forms due to combinatorial explosion. Uncovered questions constitute 34% of errors. More intelligent ways for search and candidate generation are promising future directions.

Entity linking (33%): The large-scale mining of entity surfaces forms makes entity linking a unique challenge in GRAILQA. For example, in *"After apr. the 2nd, 2009, which song was added to RB?"*, *"RB"* refers to the music video game Rock Band, while our entity linker fails to identify any entity. Also, less popular entities form another source of entity linking errors. To be specific, popularity is a very strong signal for entity disambiguation in previous datasets, but not as strong in GRAILQA because many questions are intentionally not just about very popular entities. For *"What is the belief of Mo?"* our entity linker can identify the entity mention *Mo*, and the most popular entity that has surface form *Mo* is the state of Missouri, while the question is asking about the Mo religion. Accurate entity linking on large-scale KBs is still a major challenge.

Relation mis-classification (26%): Another major type of errors comes from misclassified relations. Precisely understanding a vaguely or implicitly expressed relation is challenging, especially for zero-shot generalization. For example, question *"Which video game engine is the previous version of Frostbite 2?"* is from an unseen domain `cvg`, and the correct relation for this question should be `cvg.computer_game_engine.successor_engine`, while RANKING predicts `cvg.computer_game_engine.predecessor_engine`, which fails to capture the alignment between `successor_engine` and *"previous version"*. Besides, around 20% of relation mis-classifications are due to that the model cannot correctly determine the answer type. For question *"Name the soundtrack that Amy Winehouse features."*, which is asking about a music soundtrack, but RANKING predicts a relation associated with the answer type `music.album`.

Other (7%): The rest of the errors mainly include typing errors and function errors. For typing errors, the question can seek for information about more specific types (*e.g.*, looking for `politician` instead of just `person`), while RANKING does not explicitly handle type constraint as introducing type constraints by enumeration will result in prohibitively many candidate logical forms. A more flexible way of handling type constraint is in need. Also, sometimes RANKING can be confused by the function of a question. For instance, for question *"Name the image which appears in the topic gallery armenian rock."*, there shouldn't be any aggregation function, however, RANKING mistakenly predicts a counting function for it.

2.5.5 Transfer Learning

We also show that GRAILQA could serve as a valuable pre-training corpus for KBQA in general by pre-training RANKING on GRAILQA and testing its transferability to WEBQ, which contains naturally-occurring questions from Google search log. To adapt RANKING to WEBQ, we first convert SPARQL queries in WEBQ to our S-expression logical forms. We test three settings: Full training that fine-tunes using all training data of WEBQ, few-shot that only uses 10%, and zero-shot that directly applies models trained on GRAILQA to WEBQ. For simplicity, we assume perfect entity linking. The results are shown below. Pre-trianing on GRAILQA uniformly improves the performance, especially in the low-data regime. Most remarkably, pre-trained on GRAILQA, RANKING can achieve an F1 of 43% without *any* in-domain training on WEBQ. These results provide further supporting evidence for the quality and diversity of GRAILQA.

		Exact Match	F1
Full Training	**RANKING**	0.59	0.67
	+pre-training	**0.60**	**0.70**
Few-shot	**RANKING**	0.44	0.53
	+pre-training	**0.55**	**0.65**
Zero-shot	**RANKING**	0.00	0.00
	+pre-training	**0.35**	**0.43**

2.6 Related Work

Existing KBQA datasets. There have been an array of datasets for KBQA in recent years. WEBQ [10] is collected from Google search logs with the answers annotated via crowdsoursing. [195] later provide logical form annotation for the questions in WEBQ. Most of the questions in WEBQ are simple single-relational questions, and [168] generate more complex questions by automatically extending the questions in WEBQ with additional SPARQL statements, leading to the COMPLEXWEBQ dataset. However, such automatic extension may lead to unnatural questions. SIMPLEQ [16] is another popular KBQA dataset created by sampling individual facts from FREEBASE and annotating them as natural language questions. It therefore only contains single-relational questions. GRAPHQ [161] is the most related to ours. It proposes an algorithm to automatically generate logical forms from large-scale KBs like FREEBASE with guarantees on well-formedness and non-redundancy. We also use their algorithm for logical form generation, and develop a crowdsourcing framework which significantly improves the scalability and diversity. The aforementioned datasets are all based on FREEBASE. QALD [114] and LC-QUAD [176] are popular datasets on DBPEDIA. QALD is manually created by expert annotators, while LC-QUAD first generates SPARQL queries and unnatural canonical questions with templates and have expert annotators paraphrase the canonical questions.

Most KBQA datasets primarily operate with the i.i.d. assumption because their test set is a randomly sampled subset of all the data [10, 195, 16, 168, 176]. GRAPHQ [161] is set up to primarily test compositional generalization by splitting their training and test sets on logical forms, similarly for QALD [114]. However, none of the existing datasets supports the evaluation of all three levels of generalization. Our dataset also compares favorably to existing datasets in other dimensions such as size, coverage, diversity (Table 2.1).

Existing models on KBQA. KBQA models can be roughly categorized into semantic-parsing based methods and information-retrieval methods. The former maps a natural language utterance into a logical form that can be executed to get the final answer, while the latter directly ranks a list of candidate entities without explicitly generating a logical form. We focus on semantic-parsing methods due to their superior performance and better interpretability. Existing methods are mainly focused on i.i.d. setting and are limited in generalizability. Conventional rule-based methods [10] suffer from the coverage of their hand-crafted rules and therefore have rather limited generalizability. More recent models either employ encoder-decoder framework [68, 42, 203] to decode the logical form auto-regressively or first generate a set of candidate logical forms according to predefined templates or rules, and then match each candiate with the utterance to get the best matched one[10, 194, 34, 137, 1, 101, 102, 164, 81, 41, 64]. These models already achieve impressive results on i.i.d. dataset like WEBQ, however, on GRAPHQ that mainly tests compositional generalization, the best model can only achieve an F1 of 21.5 [164]. This demonstrates that non-i.i.d. generalization in KBQA has not drawn enough attention from existing methods.

Pre-training and non-i.i.d. generalization. The problem of non-i.i.d. generalization has drawn attention under related semantic parsing settings. [162] recognize the key role of pre-trained word embeddings [107] in cross-domain semantic parsing. Contextual embeddings like BERT are later shown to be successful for cross-domain text-to-SQL parsing [66]. Another line of work has been focusing on compositional generalization on a number of specifically-synthesized datasets [80, 73]. Concurrent to this work, [45] find that pre-trained contextual embeddings plays a more vital role in compositional generalization than specialized architectures. To the best of our knowledge, our work is among the first to demonstrate the key role of contextual embeddings like BERT at multiple levels of generalization for KBQA.

2.7 Conclusion

In this chapter, we explicitly lay out and study three levels of extrapolation using KBQA as a testbed, *i.e.*, *i.i.d.*, *compositional*, and *zero-shot*. We construct and release GRAILQA, a large-scale, high-quality KBQA dataset with 64,331 questions that can be used to evaluate all three levels of generalization. We also propose a novel BERT-based model to handle the complex KB environment. The combination of our dataset and model enables us to thoroughly examine the problem of extrapolation under diverse goals specified in natural language. In particular, we demonstrate the key role of pre-trained contextual embeddings like BERT serving as $\mathcal{M}$ in non-i.i.d. extrapolation and the advantage of discrimination and constrained decoding over unconstrained decoding. These findings help us to better understand the problem of extrapolating to new goals. We will introduce two strong methods that achieve better extrapolation in next chapters, motivated by these findings.

Chapter 3: Integrate Pre-Trained Language Models with Constrained Decoding

In this chapter, we propose a strong method, ArcaneQA, that integrates pre-trained language models and constrained decoding in a seamless way, built upon the findings from the previous chapter. Specifically, there are two key components in ArcaneQA: *contextualized encoding* and *dynamic program induction*. Dynamic program induction computes the admissible tokens for each decoding step, while contextualized encoding jointly encode the admissible tokens with the context to provide better representations of them. These two modules mutually boost each other and lead to strong extrapolation on GRAILQA.

3.1 Introduction

Modern knowledge bases (KBs) contain a wealth of structured knowledge. For example, FREEBASE [15] contains over 45 million entities and 3 billion facts across more than 100 domains, while GOOGLE KNOWLEDGE GRAPH has amassed over 500 billion facts about 5 billion entities [163]. Question answering on knowledge bases (KBQA) has emerged as a user-friendly solution to access the massive structured knowledge in KBs.

KBQA is commonly modeled as a semantic parsing problem [201, 202] with the goal of mapping a natural language question into a logical form that can be executed against the KB [10, 19, 194]. Compared with other semantic parsing settings such as text-to-SQL parsing [209, 200], where the

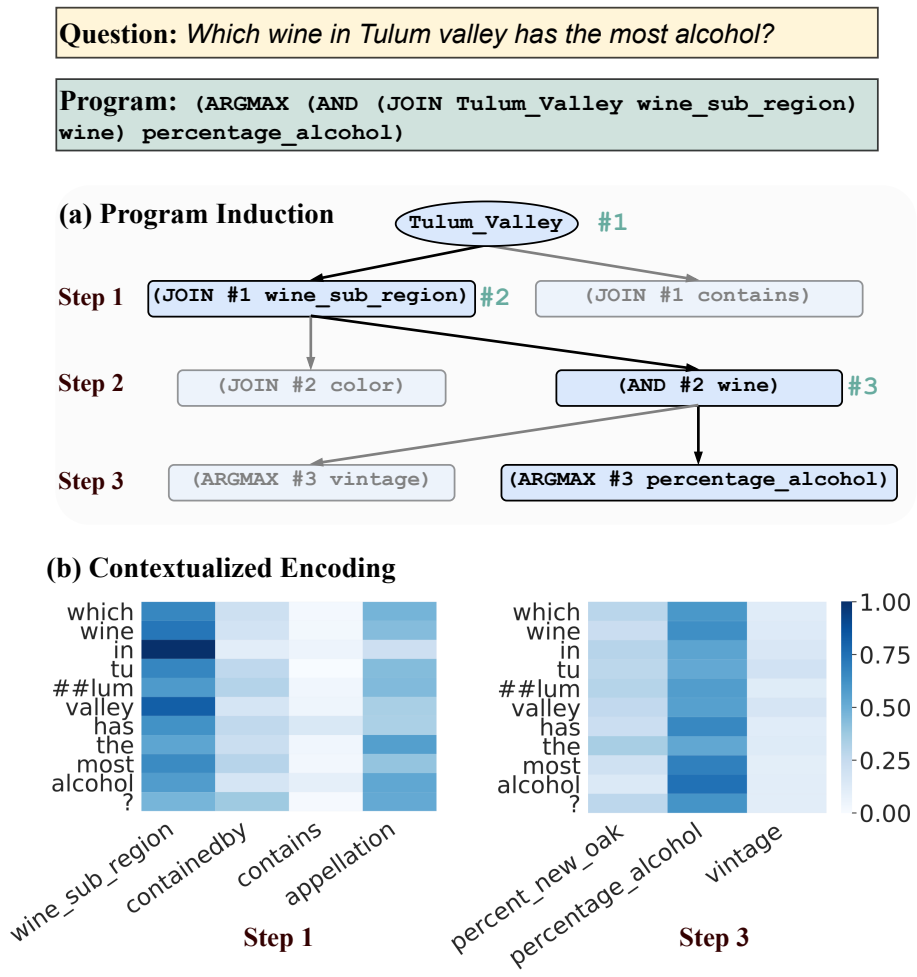

Figure 3.1: KBQA is commonly modeled as semantic parsing with the goal of mapping a question into an executable program. **(a)** A high-level illustration of our program induction procedure. The target program is induced by incrementally synthesizing a sequence of subprograms (`#1-3`). The execution of each subprogram can significantly reduce the search space of subsequent subprograms. **(b)** Alignments between question words and schema items at different steps achieved by a BERT encoder. A pre-trained language model like BERT can jointly encode the question and schema items to get the contextualized representation at each step, which further guides the search process.

underlying data is moderate-sized, the massive scale and the broad-coverage schema of KBs makes KBQA a uniquely challenging setting for semantic parsing research.

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

other semantic parsing settings [42, 178] struggle with the large vocabulary size and often generate logical forms (*i.e.*, formal queries)[14] that are not faithful to the underlying KB [170, 186]. Therefore, a candidate enumeration and ranking approach is commonly adopted for KBQA [11, 194, 1, 82, 164, 170, 193]. However, these methods have to make various compromises on the complexity of admissible logical forms to deal with the large search space. Not only does this limit the type of answerable questions, but it also leads to impractical runtime performance due to the time-consuming candidate enumeration [170]. On the other hand, schema linking,[15] *i.e.*, mapping natural language to the corresponding schema items in the KB (*e.g.*, in Figure 3.1, `wine_sub_region` is a linked schema items), is also a core challenge of KBQA. Compared with text-to-SQL parsing [66, 204, 178], the broad schema of KBs and the resulting ambiguity between schema items makes accurate schema linking more important and challenging for KBQA. Recent studies show that contextualized joint encoding of natural language questions and schema items with BERT [40] can significantly boost the schema linking accuracy [170, 27]. However, existing methods still struggle with the large search space and need to encode a large number of schema items, which is detrimental to both accuracy and efficiency.

We present ArcaneQA (Dyn**a**mic P**r**ogram Indu**c**tion **a**nd Co**n**textualized **E**ncoding for **Q**uestion **A**nswering), a *generation-based* KBQA model that addresses both the large search space and the schema linking challenges in a unified framework. Compared with the predominant ranking-based KBQA models, our generation-based model can prune the search space on the fly and thus is more flexible to generate diverse queries without compromising the expressivity or complexity of answerable questions. Inspired by prior work [42, 91, 144, 27], we model KBQA using the encoder-decoder framework. However, instead of top-down decoding with grammar-level constraints as in prior work,

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

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

which does not guarantee the faithfulness of the generated queries to the underlying KB, ArcaneQA performs *dynamic program induction* [91, 144], where we incrementally synthesize a program by dynamically predicting a sequence of subprograms to answer a question; *i.e.*, *bottom-up parsing* [30, 139]. Each subprogram is grounded to the KB and its grounding (*i.e.*, denotation or execution results) can further guide an efficient search for faithful programs (see Figure 3.1(a)). In addition, we unify the meaning representation (MR) for programs in KBQA using S-expressions and support more diverse operations over the KB (*e.g.*, numerical operations such as `COUNT/ARGMIN/ARGMAX` and diverse graph traversal operations).

At the same time, we employ pre-trained language models (PLMs) like BERT to jointly encode the question and schema items and get the contextualized representation of both, which implicitly links words to the corresponding schema items via self-attention. One unique feature to note is that *the encoding is also dynamic*: at each prediction step, only the set of admissible schema items determined by the dynamic program induction process needs to be encoded, which allows extracting the most relevant information from the question for each prediction step while avoiding the need to encode a large number of schema items. Figure 3.1(b) illustrates the contextualization of different steps via the attention heatmaps of BERT. In this example, the attention of each question word over candidate schema items serves as a strong indicator of the gold items for both steps (*i.e.*, `wine_sub_region` for step 1 and `percentage_alcohol` for step 3). The two key ingredients of our model are *mutually boosting*: dynamic program induction significantly reduces the number of schema items that need to be encoded, while dynamic contextualized encoding intelligently guides the search process.

Our main contribution is as follows: a) We propose a novel generation-based KBQA model that is flexible to generate diverse complex queries while also being more efficient than ranking-based models. b) We propose a novel strategy to effectively employ PLMs to provide contextualized

encoding for KBQA. c) We unify the meaning representation (MR) of different KBQA datasets and support more diverse operations. d) With our unified MR, we evaluate our model on three popular KBQA datasets and show highly competitive results.

3.2 Related Work

Ranking-Based KBQA. To handle the large search space in KBQA, existing studies typically rely on hand-crafted templates with a pre-specified maximum number of relations to enumerate candidate logical forms [194, 1, 82, 12, 13], which suffers from limited expressivity and scalability. For example, [194] limit the candidate programs to be a core relational chain, whose length is at most two, plus constraints. [193] additionally adopts a post-generation module to revise the enumerated logical forms into more complicated ones, however, their method still heavily depends on the candidate enumeration step. In addition, the time-consuming candidate enumeration results in impractical online inference time for ranking-based models. In contrast, ArcaneQA obviates the need for candidate enumeration by pruning the search space on the fly and thus can generate more diverse and complicated programs within practical running time.

Generation-Based KBQA. To relax the restriction on candidate enumeration, some recent efforts are made to reduce the search space using beam search [83, 29, 81], however, [83] and [29] can only generate programs of path structure, while [81] follow the query graph structure proposed by [194]. A few recent studies [91, 27] formulate semantic parsing over the KB as sequence transduction using encoder-decoder models to enable more flexible generation. [27] apply schema-level constraints to eliminate ill-formed programs from the search space, however, they do not guarantee the faithfulness of predicted programs. Similar to [91], our dynamic program induction uses KB contents-level constraints to ensure the faithfulness of generated programs, but we extend it to handle more complex and diverse questions and also use it jointly with dynamic contextualized encoding.

Using PLMs in Semantic Parsing. PLMs have been widely applied in many semantic parsing tasks, typically being used to jointly encode the input question and schema items[66, 204, 178, 142]. However, PLMs have been under-exploited in KBQA. One major difficulty of using PLMs in KBQA lies in the high volume of schema items in a KB; simply concatenating all schema items with the input question for joint encoding, as commonly done in text-to-SQL parsing, will vastly exceed PLMs' maximum input length. Existing KBQA models either use PLMs to provide features for downstream classifiers[81, 164] or adopts a pipeline design to identify a smaller set of schema items beforehand and only use PLMs to encode these identified items [170, 27], which can lead to error propagation. By comparison, ArcaneQA can fully exploit PLMs to provide contextualized representation for the question and schema items dynamically, where only the most relevant schema items are encoded at each step. More recently, [193] use T5 [135] to output a new program given a program as input, while T5's decoder generates free-formed text and does not always produce faithful programs. By contrast, ArcaneQA only uses PLMs for encoding and uses its customized decoder with a faithfulness guarantee.

3.3 Background

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

Meaning Representation for KBQA. Prior work adopt different meaning representations to represent logical forms for KBQA. For example, Yih et al.[194] use graph query, which represents a program as a core relation chain with (optionally) some entity constraints. Cai et al.[19] use λ-Calculus as their meaning representation. In this chapter, we follow [170] to use S-expressions as our meaning representation due to their expressivity and simplicity. To support more diverse

operations over the KB, we extend their definitions with two additional functions `CONS` and `TC`, which are used to support constraints with implicit entities and temporal constraints respectively. For implicit entities, consider the question *"What was Elie Wiesel's father's name?"*, whose target query involves two entities: `Elie_Wiesel` and `Male`. The entity `Male` is an implicit constraint rather than a named entity,[16] and it is used as an argument of `CONS` in the target logical form: `(CONS (JOIN people.person.children Elie_Wiesel) people.person.gender Male)`. `TC` works in a similar way, with the difference being that the constraint should be a temporal expression (*e.g.*, 2015-08-10) rather than an implicit entity.

3.4 Approach

3.4.1 Overview

The core idea of our generation-based model is to gradually expand a subprogram (*i.e.*, a partial query) into the finalized target program, instead of enumerating all possible finalized programs from the KB directly, which suffers from combinatorial explosion. There are two common strategies to instantiate the idea of gradual subprogram expansion, depending on the type of meaning representation being used. For a graph-like meaning representation, we can directly perform graph search over the KB to expand a subprogram [29, 81]. Also, we can linearize a program into a sequence of tokens and perform decoding in the token space [91, 142]. Because S-expressions can be easily converted into sequences of tokens, we choose to follow the second strategy and take advantage of the encoder-decoder framework, which has been a de facto choice for many semantic parsing tasks. Concretely, ArcaneQA learns to synthesize the target program by dynamically generating a sequence of subprograms token by token until predicting $\langle EOS \rangle$, where each subsequent subprogram is an expansion from one or more preceding subprograms (denoted as parameters in the subsequent

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

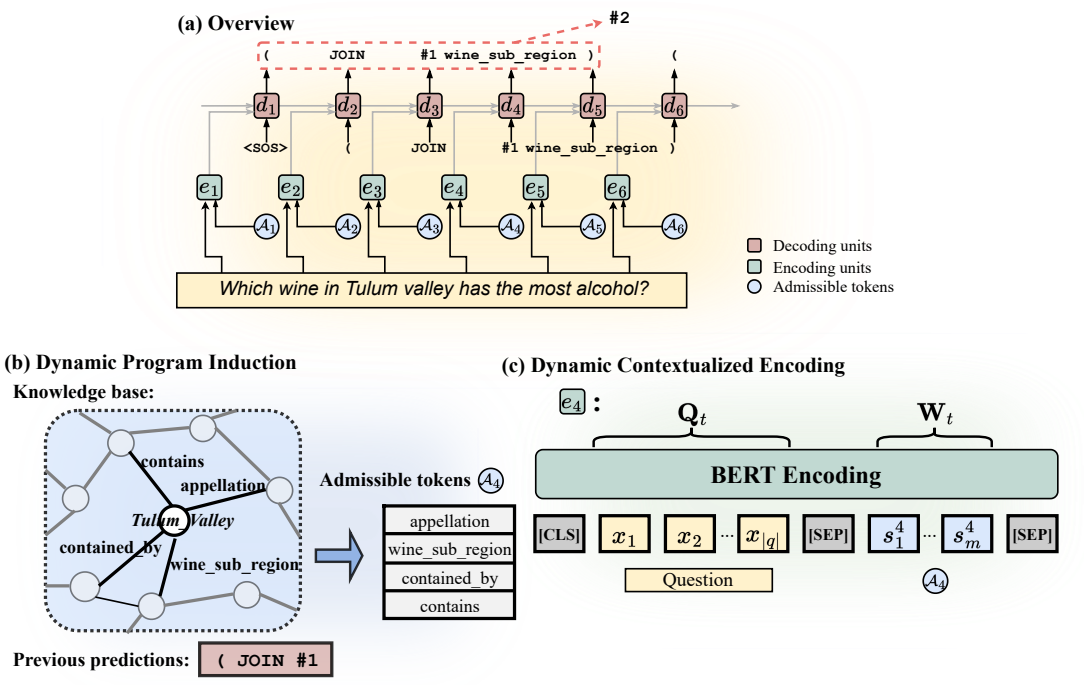

Figure 3.2: **(a)** Overview of ArcaneQA. ArcaneQA synthesizes the target program by iteratively predicting a sequence of subprograms. **(b)** At each step, it makes a prediction from a small set of admissible tokens $\mathcal{A}$ dynamically determined based on the execution of previous subprograms (for faithfulness to the KB) as well as the grammar (for well-formedness). **(c)** ArcaneQA also leverages BERT to provide dynamic contextualized encoding of the question and the admissible tokens at each step, which enables implicit schema linking and guides the dynamic program induction process.

subprogram). Formally, the goal is to map an input question $q = x_1, ..., x_{|q|}$ to a sequence of subprograms $o = o^1_1, ..., o^1_{|o^1|}, ..., o^k_1, ..., o^k_{|o^k|} = y_1, ..., y_{|o|}$, where k is the number of total subprograms and $|o| = \sum_{i=1}^{k} |o^i|$. We base ArcaneQA on Seq2Seq with attention [165, 8], in which the conditional probability $p(o|q)$ is decomposed as:

$$p(o|q) = \prod_{t=1}^{|o|} p(y_t|y_{<t}, q), \tag{3.1}$$

where each token y_t is either a token from the vocabulary $\mathcal{V}$ or an intermediate subprogram from the set $\mathcal{S}$ storing all previously generated subprograms. $\mathcal{V}$ comprises all schema items in $\mathcal{K}$, syntactic

symbols in S-expressions (*i.e.*, parentheses and function names), and the special token $\langle EOS \rangle$. $\mathcal{S}$ initially contains the identified entities in the question (*e.g.*, `#1` in Figure 2.1). Every time a subprogram is predicted, it is executed and added to $\mathcal{S}$ (*e.g.*, `#2` in Figure 2.1).

ArcaneQA builds on two key ideas: *dynamic program induction* and *dynamic contextualized encoding* (see Figure 3.2). At each decoding step, ArcaneQA only makes a prediction from a small set of admissible tokens instead of the entire vocabulary. This is achieved by the dynamic program induction framework (subsection 3.4.2), which effectively prunes the search space by orders of magnitude and guarantees that the predicted programs are faithful to the KB. In addition, we dynamically apply BERT to provide contextualized joint encoding for both the question and admissible tokens at each decoding step (subsection 3.4.3). In this way, we allow the contextualized encoding to only focus on the most relevant information without introducing noise from irrelevant tokens.

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

Table 3.1: Detailed descriptions of functions defined in our S-expressions. We extend the definitions in [170] by introducing two new functions `CONS` and `TC`. Also, we remove the function R and instead represent the inverse of a relation by adding a suffix "`_inv`" to it. Note that, for arguments in `AND` function, a class $c \in \mathcal{C}$ can also indicate a set of entities which fall into c.

3.4.2 Dynamic Program Induction

Dynamic program induction capitalizes on the idea that a complicated program can be gradually expanded from a list of subprograms. To ensure the expanded program is faithful to the KB, we query the KB with a subprogram to expand and a function defined in Table 3.1 to get a set

of admissible actions (tokens). For example, in Figure 3.2, given `#1` and the function `JOIN`, the admissible actions are defined by predicting a relation connecting to the execution result of `#1` (*i.e.*, `Tulum_Valley`), and there are only four relations to choose from (*e.g.*, `appellation` and `wine_sub_region`). Table 3.2 shows a comprehensive description of expansion rules for different functions. With these rules, ArcaneQA can greatly reduce the search space for semantic parsing over the KB dynamically. The reduced candidate space further allows us to perform dynamic contextualized encoding (subsection 3.4.3).

Within our encoder-decoder framework, this idea is implemented using constrained decoding [91, 142], *i.e.*, at each decoding step, a small set of admissible tokens from the vocabulary is determined based on the decoding history following predefined rules. The expansion rules in Table 3.2 have already comprised part of our rules for constrained decoding. In addition, several straightforward grammar rules are applied to ensure that the generated programs are well-formed. For instance, after predicting "(", the admissible tokens for the next step can only be a function name. After predicting a function name, the decoder can only choose a preceding subprogram to expand. After predicting ")", the admissible tokens for next step can only be either "(", indicating the start of a new subprogram, or "$\langle EOS \rangle$", denoting termination. The decoding process can be viewed as a *sequential decision-making process*, which decomposes the task of finding a program from the enormous search space into making decisions from a sequence of smaller search spaces.

3.4.3 Dynamic Contextualized Encoding

In semantic parsing, PLMs have typically been used to jointly encode the input question and all schema items via concatenation [66, 204, 178]. However, direct concatenation is not feasible for KBQA due to a large number of schema items. Instead of obtaining a static representation for the question and items from $\mathcal{V}$ before decoding [170, 27], we propose to do dynamic contextualized encoding at each decoding step; for each step, we use BERT to jointly encode the question and only

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

Table 3.2: A set of rules to expand a preceding subprogram given a function. The execution of the subprogram is denoted as #. These expansion rules reduce the search space significantly with a faithfulness guarantee. COUNT takes no other argument, so the only admissible token is ")".

the admissible tokens from $\mathcal{V}$. ArcaneQA's dynamic program induction vastly reduces the number of candidate tokens at each step and allows us to concatenate the question and the admissible tokens into a compact sequence:[17]

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

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

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

3.4.4 Decoding

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

$$\mathbf{h_t} = \text{LSTM}_\theta(\mathbf{h_{t-1}}, \mathbf{c_{t-1}}) \tag{3.2}$$

where our LSTM decoder is parameterized by θ.

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

$$p(y_t = s_{ti}|q, y_{<t}) = [\text{Softmax}(\mathbf{W_t h_t})]_i \tag{3.3}$$

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

$$\mathbf{a_t} = \text{softmax}(\mathbf{Q_t h_t}) \tag{3.4}$$

$$\mathbf{q_t} = (\mathbf{a_t})^T \mathbf{Q_t} \tag{3.5}$$

$$\mathbf{c_t} = [[\mathbf{W_t}]_j; \mathbf{q_t}] \tag{3.6}$$

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

3.4.5 Training and Inference

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

During inference, ArcaneQA assumes an entity linker to identify a set of entities from the question at the beginning of program induction. However, the entity linker may identify false entities. To deal with it, ArcaneQA initiate its decoding process with different hypotheses from the set of

entities. Basically, it tries out all possible combinations of the identified entities (*i.e.*, the power set of the identified entities), considering that our entity linker normally can only identify no more than two entities from a question.

3.5 Experimental Setup

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

GRAILQA [170] is a large-scale KBQA dataset that contains complex questions with various functions, including comparatives, superlatives, and counting. It evaluates the generalizability of KBQA at three levels: i.i.d., compositional and zero-shot.

GRAPHQ [161] also contains questions of diverse nature. It is particularly challenging because it exclusively focuses on non-i.i.d. generalization.[18]

WEBQSP[195] is a clean subset of WEBQ [10] with annotated logical forms. All questions in it are collected from Google query logs, featuring more realistic and complicated information needs such as questions with temporal constraints.

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

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

Models for Comparison. We compare ArcaneQA with the previous best-performing models on three different datasets. For GRAILQA and WEBQSP, the state-of-the-art model is **RnG-KBQA** [193].

[18] GRAPHQ originally uses FREEBASE (version 2013-07) as their KB, while GRAILQA and WEBQ use FREEBASE (version 2015-08-09). In [170], programs in GRAPHQ are converted into the corresponding FREEBASE 2015-08-09 version, and we will use this version in our experiments.

Though RnG-KBQA uses T5 to decode the target program as unconstrained sequence transduction, it still heavily depends on candidate enumeration as a prerequisite. Therefore, it is not a generation-based model like ours. **ReTraCk** [27] is the state-of-the-art generation-based model on GRAILQA which poses grammar-level constraints to the decoder to generate well-formed but unnecessarily faithful programs. For GRAPHQ, the ranking-based model **SPARQA** [164] has achieved the best results so far. It uses BERT as a feature extractor for downstream classifiers. In addition to the state-of-the-art models, we also compare ArcaneQA with **BERT+Transduction** and **BERT+Ranking** [170], which are two baseline models on GRAILQA that enhance a vanilla Seq2Seq model with BERT to perform generation and ranking respectively.

Implementation. Our models are implemented using PyTorch and AllenNLP [49]. For BERT, we use the bert-base-uncased version provided by HuggingFace.

3.6 Results

3.6.1 Overall Evaluation

We show the overall results in Table 3.3. ArcaneQA achieves the state-of-the-art performance on both GRAPHQ and WEBQSP. For GRAPHQ, there are 188 questions in GRAPHQ's test set that cannot be converted into FREEBASE 2015-08-09 version, so we treat the F1 of all those questions as 0 following [170], while the numbers in the parentheses are the actual F1 on the test set if we exclude those questions. ArcaneQA significantly outperforms the prior art by over 10%. The improvement over SPARQA shows the advantage of using PLMs for contextualized joint encoding instead of just providing features for ranking. On both WEBQSP and GRAILQA, ArcaneQA also achieves the best performance or performs on par with the prior art in terms of F1. It outperforms ReTraCk by 4.3% and 1.9% (using the same entity linking results) on WEBQSP and GRAILQA respectively, suggesting that ArcaneQA can more effectively reduce the search space with dynamic program

Model	Overall		I.I.D.		Compositional		Zero-shot	
	EM	F1	EM	F1	EM	F1	EM	F1
QGG* [81]	–	36.7	–	40.5	–	33.0	–	36.6
BERT+Transduction* [170]	33.3	36.8	51.8	53.9	31.0	36.0	25.7	29.3
BERT+Ranking* [170]	50.6	58.0	59.9	67.0	45.5	53.9	48.6	55.7
ReTraCk [27]	58.1	65.3	84.4	87.5	61.5	70.9	44.6	52.5
RnG-KBQA* [193]	61.4	67.4	78.0	81.8	55.0	63.2	56.7	63.0
ArcaneQA*	58.8	67.2	77.8	81.6	58.0	66.1	50.4	61.8
RnG-KBQA [193]	**68.8**	**74.4**	**86.2**	**89.0**	63.8	71.2	**63.0**	**69.2**
ArcaneQA	63.8	73.7	85.6	88.9	**65.8**	**75.3**	52.9	66.0
w/o contextualized encoding	49.7	59.1	77.6	82.1	50.5	59.4	36.5	48.5

(a) GRAILQA

Model	F1
UDEPLAMBDA [137]	17.7
PARA4QA [42]	20.4
SPARQA [164]	21.5
BERT+Ranking [170]	25.0 (27.0)
ArcaneQA	**31.8** (**34.3**)
w/o contextualized encoding	20.7 (22.4)

(b) GRAPHQ

Model	F1
NSM [91]	69.0
KBQA-GST [82]	67.9
TextRay [12]	60.3
QGG [81]	74.0
ReTraCk [27]	71.0
CBR [35]	72.8
RnG-KBQA [193]	**75.6** (74.5$^\sharp$)
ArcaneQA	**75.6** (**75.6**$^\sharp$)
w/o contextualized encoding	68.8

(c) WEBQSP

Table 3.3: Overall results on three datasets. ArcaneQA follows entity linking results from previous methods (*i.e.*, RnG-KBQA's results on GRAILQA, QGG's results on WEBQSP, and Gu et al. [170]'s results on GRAPHQ) for fair comparison. Model names with * indicate using the baseline entity linking results on GRAILQA. $^\sharp$ In addition to using WEBQSP's official evaluation script, which sometimes considers multiple target parses for a question, we also report the performance when only the top-1 target parses are considered.

induction compared with ReTraCk's grammar-based decoding. Also, our model performs on par with the previous state-of-the-art RnG-KBQA (*i.e.*, same numbers on WEBQSP, while 0.7% lower on GRAILQA). However, ArcaneQA under-performs RnG-KBQA in EM on GRAILQA. The overall EM of ArcaneQA is lower than RnG-KBQA by 5%, and the gap on zero-shot generalization is even larger (*i.e.*, around 10%), despite the comparable numbers in F1. This can be explained by that ArcaneQA learns to predict a program in a more flexible way and may potentially find some novel structures. This may further be supported by the observation that ArcaneQA performs better than RnG-KBQA on compositional generalization, which requires KBQA models to generalize to unseen

query structures during training. Overall, the results demonstrate ArcaneQA's flexibility in handling KBQA scenarios of different natures.

3.6.2 In-Depth Analyses

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

Dynamic Program Induction. One vanilla implementation of ArcaneQA without dynamic program induction is BERT+Transduction, *i.e.*, its search space and vocabulary during decoding is independent of previous predictions. As shown in Table 3.3a, when using the same entity linking results, ArcaneQA outperforms BERT+Transduction by 30.4% in overall F1 and is twice as good on zero-shot generalization. One major weakness of BERT+Transduction is that it predicts many programs that are not faithful to the KB, executing which will lead to empty answers. Note that post-hoc filtering by execution [179] can only help to a limited degree due to the KB's broad schema, while this type of mistake is rooted out in ArcaneQA by design.

Different from our search space pruning achieved with dynamic program induction, ranking-based models such as BERT+Ranking prunes unfaithful programs from their search space by ranking a set of faithful programs enumerated from the KB. These models typically make compromises on the complexity and diversity of programs during candidate enumeration. We break down the performance of ArcaneQA on GRAILQA's validation set in terms of question complexity and function types and show the fine-grained results in Table 3.4. The comparison with BERT-Ranking demonstrates the scalability and flexibility of our dynamic program induction. We also compare with RnG-KBQA, which adopts exactly the same candidate enumeration module as BERT+Ranking, but it is enhanced with a T5-based revision module to edit the enumerated programs into more diverse ones. We observe that RnG-KBQA performs uniformly well across different programs except for programs

with superlative functions (*i.e.*, ARGMAX/ARGMIN), *i.e.*, the F1 of it is lower than ArcaneQA by over 50%. This is because in their candidate generation step, there is no superlative function enumerated. Despite the effectiveness of their T5-based revision, their performance still heavily depends on the diversity of candidate enumeration, which restricts the flexibility of their method.

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

Table 3.4: Fine-grained results (EM/F1) on GRAILQA's dev set. **None** denotes programs with only AND and JOIN.

Dynamic Contextualized Encoding. To show the key role of dynamic contextualized encoding, we use GloVe [130] to provide non-contextualized embeddings for both questions and tokens in $\mathcal{V}$. We fix GloVe embeddings during training to make the model less biased to the training distribution [170] for GRAILQA and GRAPHQ, which address non-i.i.d. generalization, while for WEBQSP, we also update the word embeddings during training. Results in Table 3.3a show the importance of dynamic contextualized encoding, *i.e.*, without contextualized encoding, the overall F1 decreases by 14.6%, 11.1%, and 6.5% on three datasets respectively. We also notice that dynamic contextualized encoding is more critical for non-i.i.d. generalization, *i.e.*, on GRAILQA the F1 on i.i.d. generalization only decreases by 6.8%, while it decreases by 15.9% and 17.5% on compositional and zero-shot generalization. Without contextualized encoding, identifying the correct schema items from the KB

in non-i.i.d. setting is particularly challenging. Schema linking powered by dynamic contextualized encoding is the key to non-i.i.d. generalization, which is a long-term goal of KBQA.

3.6.3 Efficiency Analysis

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

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

3.7 Conclusion

Motivated by our findings in Chapter 2, we present a novel model, ArcaneQA, that combines pre-trained language models and constrained decoding in a unified way. ArcaneQA simultaneously addresses the large search space and schema linking challenges in KBQA with dynamic program induction and dynamic contextualized encoding. Experimental results on several benchmarks, including GRAILQA, demonstrate the advantages of ArcaneQA in both effectiveness and efficiency. However, we find that the constrained decoding method remains sub-optimal in the zero-shot setting, despite the improvement over baseline methods. In the next chapter, we will explore how a discrimination-based framework can further enhance performance for non-i.i.d. extrapolation.

Chapter 4: A General Discrimination Framework for Enhanced Extrapolation

In this chapter, we further propose a general framework, Pangu, that is compatible with various types of language models, including the recent large language models (LLMs). Our design choice is directly guided by the insight that language models excel as $\mathcal{M}$ while discrimination works best for f. Specifically, we propose using language models to perform ranking or discrimination rather than generation, as proposed in the first solution. Our framework is compatible with any kind of language models, be it encoder-only, decoder-only, or encoder-decoder. In addition, we present the first solution that successfully adapts LLMs to KBQA, which involves a massive action space that poses a challenge to the context limit of language models. Our method achieves strong results, remaining one of the top methods on GRAILQA even two years later. Upon close investigation, we find that the main reason for the strong performance is that generation-based methods tend to overfit to the training set, which leads to suboptimal extrapolation, whereas discrimination models capture a more robust relationship between natural language goals and target actions, highlighting a generator-discriminator gap.

4.1 Introduction

Language models (LMs) such as BERT [40], GPT-3 [116], and Codex [117] have demonstrated an extraordinary capacity in understanding and generating both natural language [108, 93] and generic programs (*e.g.*, Python) [89, 67, 6]. The recent release of ChatGPT is elevating this paradigm

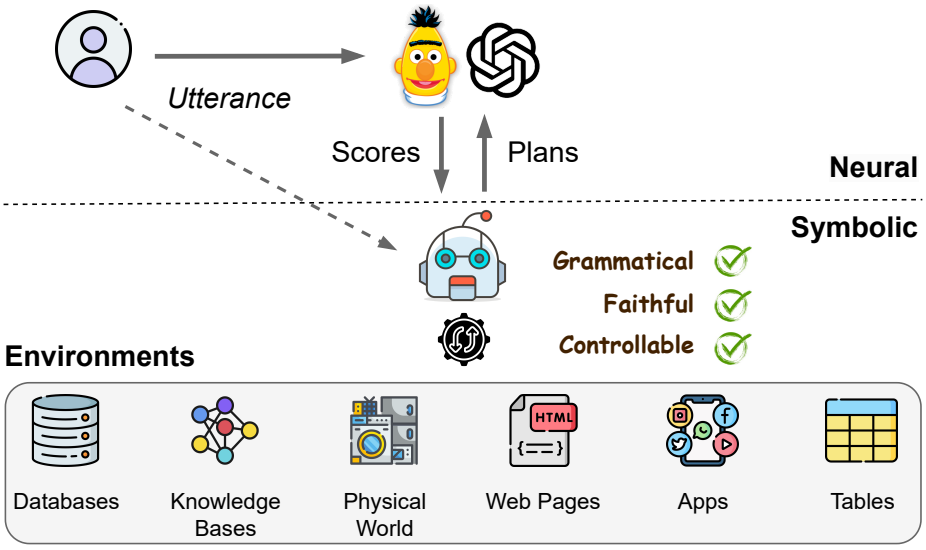

Figure 4.1: A schematic illustration of the proposed framework, Pangu, where a symbolic agent interacts with the target environment to propose candidate plans, and a neural LM evaluates the plausibility of each plan. The agent searches the environment to incrementally construct the plans, and the LM guides the search process.

to a new level.[19] 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 [149, 2]). 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*

[19] `chat.openai.com`

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) concepts to contexts [25], therefore becomes the fundamental problem. More precisely, we need to produce a *plan* (also called a *program* when described using a programming language) that can be executed in an environment to achieve the desired effects of the corresponding language request. 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 [33, 165] and generate the plans/programs in an autoregressive fashion [186, 193, 181, 155]. A core thesis of this chapter 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* (*i.e.*, conforming to the grammar of the planning language) and *faithfulness* (*i.e.*, executable in the environment) of the generated plans. 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 [199, 38] or constrained decoding [142, 148, 173]. 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 chapter 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 4.1).[20] Pangu consists of a symbolic agent and a neural LM working in a concerted way. 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 [170]. We

[20]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.

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 [161]. It provides *unprecedented uniformity* for using LMs—one can easily plug encoder-only LMs, encoder-decoder LMs, or decoder-only LMs into Pangu. These results highlight the remarkable effectiveness and flexibility of Pangu and validate the proposal of using LMs for discrimination instead of generation.

4.2 Related Work

Generation for Grounded Language Understanding. The Seq2Seq framework [165, 8] 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 the LM 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 [66, 178, 186]. Such methods rely on the LM to understand the interplay between 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 [142, 151] or even faithfulness [91, 173]. 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 comparison is discussed in §4.5.3.

Few-Shot Grounded Language Understanding with LLMs. Large language models (LLMs) [116, 117] have demonstrated strong few-shot learning capabilities in various tasks, from writing programs to query structured and unstructured data [6, 136, 32], interacting with online websites [54, 112], to generating procedural plans and guiding embodied agents in virtual environments [154, 2, 145, 156]. 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. 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 [2], 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.

Bottom-Up Semantic Parsing. Our instantiation of Pangu on KBQA is closely connected to bottom-up semantic parsing, particularly SmBoP [139], 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 [85]. 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 [194, 170, 193]. Because they try to enumerate all candidate plans up front, the maximum

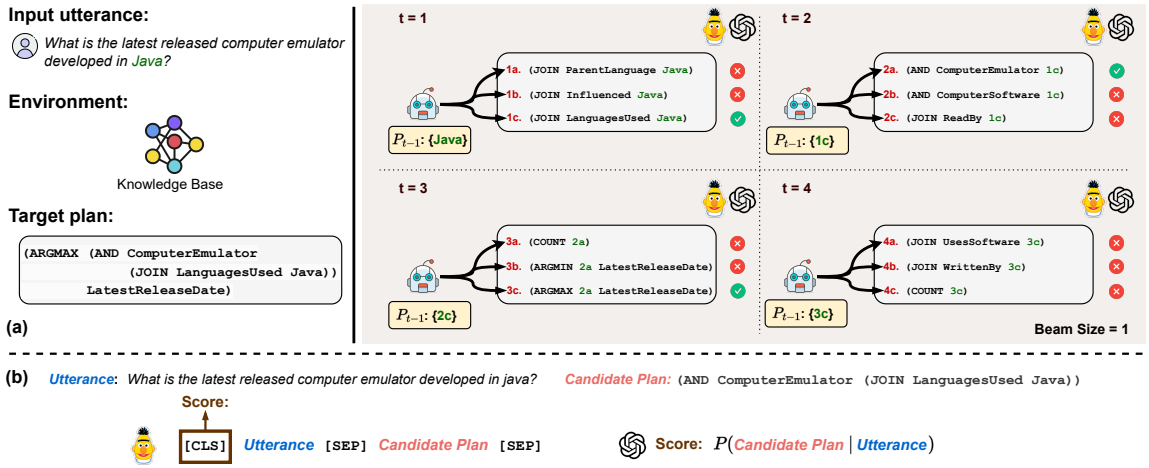

Figure 4.2: (a) An illustration of how an agent collaborates with an LM to incrementally produce a complex target plan over a KB using beam search (beam size = 1 in this example). At each step, the agent enumerates a set of valid plans based on the current plans and the environment. An LM then scores 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.

plan complexity is bound to be small. Our adaptive search process allows for flexible construction of more complex plans.

4.3 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 [149], or a virtual agent that orchestrates API calls of different web services [4] or traverses a database/knowledge base (KB) [200, 171]. 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.

4.3.1 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 [15] 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 [171], 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 [170] for its compactness. An example is shown in Figure 4.2.

Algorithm 1: PANGU

Input: utterance q, initial plans P_0, environment E
$t \leftarrow 1$;
while *True* **do**
    // AGENT PROPOSES PLANS
    $C_t \leftarrow$ **Candidate-Plans**(P_{t-1}, E)
    // LM SCORES AND PRUNES PLANS
    $P_t \leftarrow$ **Top-**$K(q, C_t)$
    if ***Check-Termination()*** = *True* **then**
        return top-scored plan
    $t \leftarrow t + 1$

4.3.2 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 [173] (Figure 4.2). 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 [85]. 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 4.2 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.

4.3.3 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: $s(u, c) \to \mathbb{R}$, 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 4.2(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 4.2(b)). For encoder-decoder LMs like T5, we follow [211] to feed both the utterance and the candidate plan to the encoder and use the decoding probability over an unused token 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 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) 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.

4.3.4 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.

4.3.5 Learning

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

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 4.2). Fine-tuning proceeds with beam search similar to the test-time behavior, but with bottom-up teacher forcing [185, 139], *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 [23].

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.

4.4 Experimental Setup

4.4.1 Datasets

We experiment with three KBQA datasets of different scale and nature.

GRAILQA [170] 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 and aggregation functions.

GRAPHQ [161] 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 [173], which maps the original dataset from FREEBASE 2013-07 to FREEBASE 2015-08-09.

WEBQSP [195] 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 [10] with program annotations.

4.4.2 Baselines

We mainly compare Pangu with state-of-the-art baselines that use LMs as a generative model, including ArcaneQA [173], TIARA [151], DecAF [198], and RnG-KBQA [193]. 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 language models to do different jobs (*i.e.*, retrieval/ranking/decoding). In addition, we also compare with UnifiedSKG [186]. 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 a large LM. Compared with the baselines, Pangu requires no

Model	Overall		I.I.D.		Compositional		Zero-shot		Dev Overall	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
QGG [81]	–	36.7	–	40.5	–	33.0	–	36.6	–	–
BERT+Ranking [170]	50.6	58.0	59.9	67.0	45.5	53.9	48.6	55.7	–	–
ReTraCk [27]	58.1	65.3	84.4	87.5	61.5	70.9	44.6	52.5	–	–
RnG-KBQA [193]	68.8	74.4	86.2	89.0	63.8	71.2	63.0	69.2	71.4	76.8
ArcaneQA [173]	63.8	73.7	85.6	88.9	65.8	75.3	52.9	66.0	69.5	76.9
Uni-Parser [99]	69.5	74.6	85.5	88.5	65.1	71.1	64.0	69.8	70.8	76.5
TIARA [151]	73.0	78.5	**87.8**	**90.6**	69.2	76.5	68.0	73.9	75.3	81.9
DecAF [198]	68.4	78.7	84.8	89.9	73.4	81.8	58.6	72.3	–	81.4
UnifiedSKG w/ T5-3B [186]	–	–	–	–	–	–	–	–	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 [137]	17.7♯
PARA4QA [43]	20.4♯
SPARQA [164]	21.5♯
BERT+Ranking [170]	27.0
ArcaneQA [173]	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 [81]	74.0
ReTraCk [27]	71.0
CBR [36]	72.8
Program Transfer [22]	76.5*
RnG-KBQA [193]	75.6
ArcaneQA [173]	75.6
Uni-Parser [99]	75.8
TIARA [151]	76.7
DecAF [198]	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 4.1: 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 [173], which is a slightly smaller subset.

extra parameter, no modification to the LM, and no need to combine multiple LMs. Pangu provides unprecedented uniformity of using LMs of different nature.

4.4.3 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.[21] We randomly sample 10/100/1000 training examples from each dataset and use that as the pool for dynamic retrieval. 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.

4.5 Results

4.5.1 Main Results

Fine-tuning results. The main results are shown in Table 4.1. 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 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

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

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.

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% only with 10 training examples. Though there is still a gap to the fine-tuning results, it is still impressive, especially considering the massive meaning space of the KB. On GRAPHQ, Pangu with Codex even outperforms ArcaneQA with 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.

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$^{\triangle}$	`(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$^{\triangle}$	`(JOIN computer.software.software_genre m.05vvqy) (AND computer.software #0)`

Table 4.2: Two representative examples that Pangu succeeds while ArcaneQA fails, both w/ BERT-base. $^{\triangle}$ denotes the original order of the decoder's output. The first incorrect token predicted by ArcaneQA is marked in red.

4.5.2 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 4.3 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.

4.5.3 Pangu vs. Constrained Decoding

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

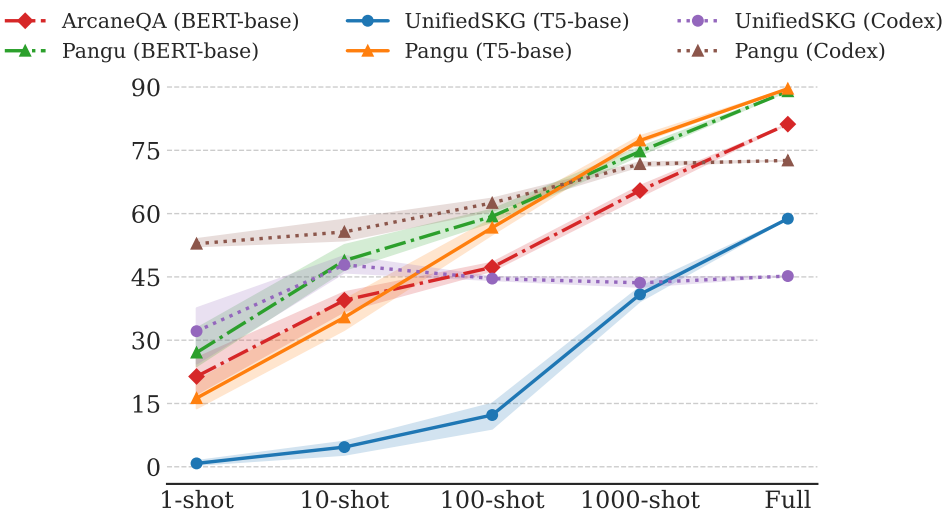

Figure 4.3: Sample efficiency results. We conduct three runs with different training examples and show the mean EM; shaded areas denote max/min.

leads to catastrophic errors (Table 4.2). 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 4.4, where Pangu's output probability distributions are consistent across programs seen and unseen in training. While for ArcaneQA, there is a drastic shift from seen to unseen. This is also consistent with the finding that autoregressive models tend to overfit seen structures during training by [14]. It makes non-i.i.d. generalization more difficult.

4.6 Conclusion

In this chapter, we propose to capitalize on the discriminative ability of language models (LMs) instead of their generative ability for grounded language understanding. Building on this proposal, we

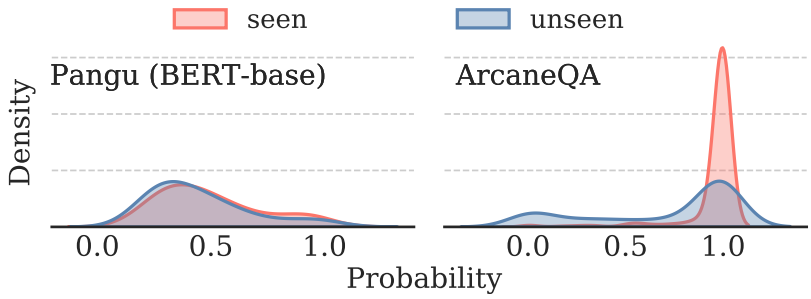

Figure 4.4: Distribution of the probabilities assigned to predicted programs that are seen and unseen during training. We use kernal density smoothing for better visualization, so the x-axis goes over 1.0.

design 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 sample-efficient extrapolative agentic 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 slower 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.

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 [184].

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 [91].

Finally, in this chapter, one important merit of Pangu, controllability, is under-explored, because it is not very necessary for KBQA. While for tasks like text-to-SQL parsing, controllability is a highly desirable property. Intruders may manipulate text-to-SQL models to launch database attacks via SQL injection [129]. 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 prediction will lead to a malicious operation several steps later. In the future, we will explore Pangu's controllability on more different tasks.

Part II: Facilitating Goal-Policy Mapping through Natural Language Prior

Chapter 5: Enhance f with Language-Conditioned Policy

In the previous chapters, we have focused mainly on the evaluation presented in Chapter 2. A key limitation in our evaluation is that KBQA mostly focuses on direct mapping: the goal descriptions operate as a similar abstraction level as the action space. However, this property may not hold in many real-world scenarios. For example, you may simply instruct your robot to "*make a pizza*," while the robot has to first figure out how to decompose this goal into subgoals. In this chapter, we relax this assumption and take it a step further by fully leveraging the autonomy of large language models (LLMs). Specifically, LLMs are proficient with using natural language as a vehicle of thought (*i.e.*, CoT reasoning [184]). Capitalized on this intuition, we focus on providing the LLM with necessary scaffolding to interface complex environments like the KB and database, and fully leverage the LLM's own decision-making capacity, which shows even stronger performance in extrapolation compared with Pangu when using no training data. This aligns with the modern paradigm of language agents [160].

5.1 Introduction

Large language models (LLMs) have demonstrated a human-like mastery over text [118, 119, 175, 70]. However, the true ambition of AI extends well beyond the realm of text. The goal is to ultimately empower LLMs to act as generalist language agents that can aid humans across the multitude of complex real-world tasks [191, 141, 97], which often involve handling complex environments, be it

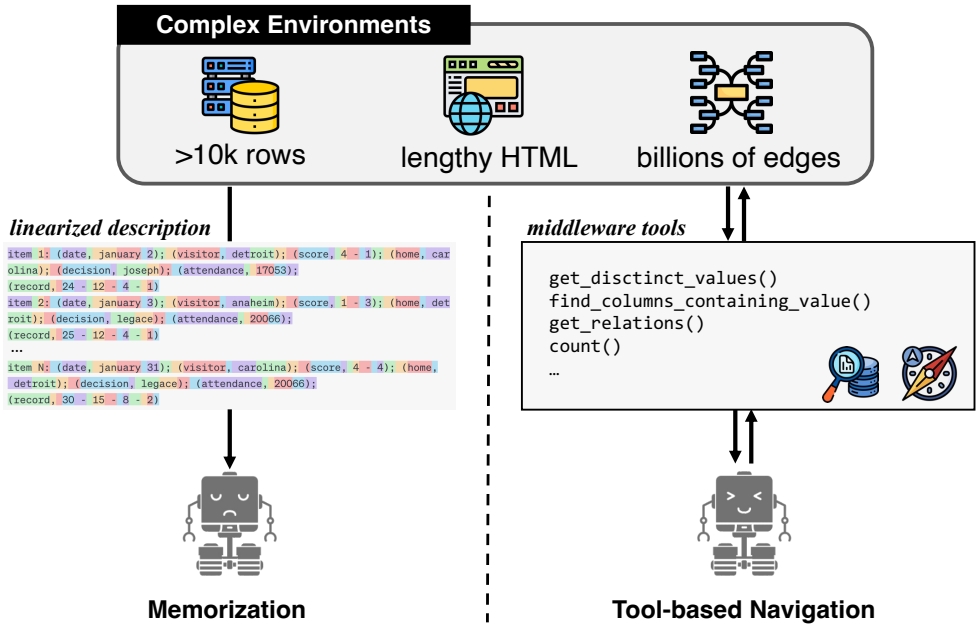

Figure 5.1: (*left*) When an LLM engages with a complex environment, it can develop an understanding by fitting the environment's description (*i.e.*, linearized tokens) into its short-term memory (*i.e.*, the LLM's input window). However, this method encounters drastic scalability issues as the complexity of the environment grows. (*right*) Another option is to furnish the LLM with a set of tools that assist it in actively engaging with the environment and acquiring the necessary information.

browsing complex webpages [39], managing vast databases with millions of entries [86], or querying huge KBs [169].

For LLMs to effectively serve as agents that ground human instructions accurately within the environment, they must develop a robust understanding of the environment. The most direct method to achieve it is to linearize the environment into a sequence of tokens that fit into the LLM's short-term memory (*i.e.*, its input window) and have the LLM process the environment based on the linearized description [167, 150, 97]. However, such a method faces steep challenges in scaling to more complex environments. Also, discrete token descriptions may not reflect the most natural perception of the environment. Recent work has explored using tools to extend the boundary of the LLM's capacity [87, 134, 141]. The core idea is that LLMs can actively decide a proper tool to use,

using language as a powerful vehicle of thought [160]. Intuitively, we can also equip the LLM with tools that enable navigating complex environments, so that the LLM can proactively invoke different tools to explore the environment, thus circumventing limitations posed by its short-term memory (Figure 5.1). However, this promising paradigm has been thus far underexplored. In this chapter, we aim to delve into this paradigm and answer an intriguing question: ***How effectively can LLMs handle complex environments with the aid of tools?***

Answering this question requires equipping the LLM with a suite of tools designed to meet a wide range of needs within the target environment. In this chapter, we carefully develop such tailored tools for two exemplar complex environments, *i.e.*, databases and knowledge bases (KBs). Unlike readily available Web APIs [134] used in prior research, our tools have to be manually invented from scratch. In crafting these tools, we capitalize on the intuition of human information-gathering behaviors—such as performing keyword searches to identify a relevant database column or investigating the connections of a KB entity—to fulfill complex tasks in these environments (Section 5.3.1). Ideally, these tools are designed to function as a *middleware* layer between the LLM and the environment, shielding the LLM from environmental complexity. With these specialized tools, we propose two novel schemes to enable the LLM to more accurately orchestrate its internal reasoning and tool usage: *error feedback* and *decoupled generation* (Section 5.3.2). The combination of the crafted tools and the tool-use schemes allows the LLM to actively explore the environment and ground human instructions into accurate actions.

We evaluate different LLMs on benchmarks featuring complex tasks over the target environments, including a newly curated benchmark for the KB. The results are revealing: *LLMs equipped with customized tools demonstrate a significant enhancement in their ability to engage with complex environments, markedly surpassing the prior art.* In particular, despite its simplicity, such a middleware layer allows GPT-4 [118] to achieve **2.8**$\times$ the performance (*i.e.*, 38.3% vs. 13.8%) of the best

baseline in tasks requiring access to database content and **2.2**$\times$ (*i.e.*, 59.3% vs. 27.1%) in KB tasks. Our findings underscore the integral role of tool augmentation in enabling LLMs to handle complex environments.

Our main contributions are as follows: a) We develop a new framework with customized tools for two complex environments, to investigate the role of tools in handling complex environments with LLMs; b) We evaluate six different LLMs on our carefully chosen benchmarks for a comprehensive analysis; c) Our analysis highlights a critical takeaway: augmenting LLMs with tools is crucial for successfully tackling complex environments, opening new possibilities to progress LLMs as generalist language agents for practical applications.

5.2 Related Work

Interface Complex Environments with LLMs. Existing methods that feed the environment directly into the LLM for grounding [26] would fail in complex environments due to scalability issues. Specifically, these methods process the environment by linearizing it into discrete tokens [66, 150, 198, 97, 167, 156]. However, linearizing expansive environments like databases with millions of entries [86] or lengthy webpage HTML code [39] can often exceed an LLM's input length constraints. Alternative studies bypass the LLM's direct interaction with complex environments by generating ungrounded draft plans for post-processing grounding [88, 115] or by using the LLM to assess grounded plans created via predefined rules [169]. Such strategies do not fully utilize the LLMs' innate reasoning potential in actively navigating complex environments. In this chapter, we explore a new paradigm where we can bypass these issues by equipping LLMs with a suite of comprehensive tools to actively gather necessary information about the environment upon demand, leveraging the LLMs' inherent reasoning capabilities.

Tool Learning. Tools are essential for enhancing the capabilities of LLMs [141, 133, 106, 58]. Existing research, such as ToolLLM [134] and API-Bank [87], focuses on open-domain applications with a wide array of readily available RESTful APIs. In contrast, this chapter specifically aims to study the potential of tools in augmenting LLMs to effectively execute tasks within complex environments, where we carefully craft the specialized tools for different environments by ourselves. In addition, research focusing on RESTful APIs typically displays shallow reasoning, while practical tasks within a complex environment typically entail a long sequence of actions (*e.g.*, querying a KB or browsing a webpage). To enable tool use in more intricate settings within a more specific complex environment, StructGPT [71] employs a predefined sequence of tool invocations; Chameleon [100] functions in an open-loop setting where the LLM directly produces a sequence for tool usage before any execution occurs. Both of them fail to seamlessly integrate the reasoning capacity of the LLM with the use of tools. In this chapter, we propose two novel schemes—*error feedback* and *decoupled generation* to more seamlessly and accurately orchestrate the LLM's internal reasoning and tool usage.

5.3 Middleware for LLMs

We equip LLMs with a suite of tools specifically tailored to support an extensive variety of operations and cater to the diverse needs within a complex environment $\mathcal{E}$. We call these tools middleware, as they can serve as a feature-rich middle layer between the LLM and $\mathcal{E}$, abstracting the LLM from having to directly interact with all of its intricacies (Section 5.3.1). Furthermore, to fully unleash the inherent reasoning capabilities of LLMs in invoking proper tools, we propose two novel schemes to enhance tool use accuracy: *error feedback*, which provides concrete tool use error information and expects the LLM to correct the error autonomously, and *decoupled generation*, where the LLM's reasoning steps and tool use are separated for better controllability (Section 5.3.2).

This unified framework allows us to reliably investigate the potential of LLMs in handling complex environments with the aid of tools.

Figure 5.2: The LLM is equipped with an array of tools to facilitate its engagement with complex environments (*e.g.*, a KB here). (a) The LLM may produce invalid actions (marked in pink). This can be mitigated by prompting it with an error message that encourages a reattempt (corrected action marked in green). (b) Alternatively, we can have the LLM first generate a thought, then predict an action based on it in a separate context (marked in blue), and finally insert the action back to the original context. Text marked in yellow are input from the environment.

5.3.1 Tools for Complex Environments

To evaluate the potential of LLMs in handling complex environments when equipped with tools, we need to first carefully craft the necessary tools for the environments. These tools should meet two essential criteria: 1) They should offer comprehensiveness, encompassing a broad spectrum of operations and needs. Broad coverage of tools is crucial for maximizing the potential of LLMs in planning. 2) The tools should prioritize ease of use, enabling the LLM to invoke them mostly with straightforward slot filling, thus shielding the LLM from the implementation details of the tools.

Databases In production scenarios, databases typically feature dozens of tables, with each table containing thousands of rows or more. A key task in such environments is performing data analysis through SQL queries. To bridge the gap between natural language instructions and SQL, LLMs are employed to automate the generation of SQL queries (*i.e.*, text-to-SQL parsing [200, 86]). To support the LLM in crafting complex SQL queries, we introduce a set of specialized tools designed for interaction with intricate databases. These tools are divided into two main categories: navigational and functional. Navigational tools help the LLM to explore the environment (*e.g.*, `get_distinct_values()` and `find_columns_containing_value()`), while functional tools help check each SQL clause composed by the LLM. For example, `where()` verifies the legality of the WHERE clause and determines if the specified conditions can match any entries in the database. In total, we craft 12 tools for databases. The development of these tools is grounded in our domain expertise in SQL and databases.

KBs Modern KBs, such as FREEBASE [15], are vast repositories storing billions of facts as triples $\langle h, r, t \rangle$. These KBs cover a wide array of domains and support complex information-seeking tasks, including answering questions that require multi-hop reasoning. To support the LLM in engaging the extremely massive KB environments, we also devise a toolset tailored for KBs. Similarly, tools

for KBs also include navigational tools and functional tools. The navigational tools facilitate efficient exploration of the KB by the LLM (*e.g.*, `get_relations()` and `get_attributes()`), while the functional tools support the LLM in executing precise operations, such as counting and intersecting two sets (*e.g.*, `intersection()` and `count()`). Both are critical for completing complex reasoning tasks on KB. A key concept in tools for KBs is a *variable*, representing a set of entities and typically generated as an intermediate result through the execution of functions like `get_neighbors()` or `intersection()`. The use of variables facilitates multi-hop reasoning across KBs, as it enables the natural linkage of a sequence of tool executions. In total, we implement 7 tools for KBs. Our design of KB tools tightly adheres to the common needs in knowledge base question answering (KBQA) [170, 21].

5.3.2 Reasoning with Tools

We first choose ReAct [191] to serve as the backbone of our reasoning framework, in which the LLM can proactively decide which tool to use based on its own chain-of-thought [184]. Based on this backbone, we propose two novel schemes to improve the accuracy of tool use, which is critical for complex tasks where successful tool use necessitates careful tool selection and precise argument assignment. Unlike existing methods relying on human-defined workflows that follow fixed-order tool usage [71], our framework allows the LLM autonomy in proactively determining tool selection using CoT.

Formally, at each step t, the LLM makes predictions following a policy that maps a current context to an output: $\pi : c_t \rightarrow \hat{a}_t$, where

$$c_t = (\hat{a}_1, o_1 \cdots, \hat{a}_{t-1}, o_{t-1})$$

$$\hat{a}_t = r_t \oplus a_t$$

$\hat{a}_t$ is the concatenation of a rationale r_t (*i.e.*, a thought in CoT) and a concrete tool use a_t (*e.g.*, in Figure 5.2, $\hat{a}_1$ is the concatenation of **`Thought 1`** and **`Act 1`**), while o_t is an observation from the environment (*i.e.*, the execution result of a_t). In ReAct, the LLM jointly decodes $\hat{a}_t$ based on c_t for each step. However, originally designed for simpler tools like the Wikipedia Search API, the naive ReAct framework is more susceptible to producing an invalid a_t that is unfaithful to r_t when applied to more nuanced tool usage. We propose two simple strategies to remedy this issue. The first strategy is to simply amplify ReAct by providing detailed *error feedback* in case of incorrect tool usage by the LLM, followed by a prompt to retry based on these messages (see Figure 5.2(a)).[22] This relies on the LLM's capacity for self-correction through feedback [50, 28], which may not always be reliable when the underpinning LLM is weak, potentially leading to the repetition of the same mistakes [51]. Additionally, we present *decoupled generation*, where the LLM's policy π is split into two sequential phases (*i.e.*, $\pi \propto \pi_1 \circ \pi_2$), allowing for more nuanced control of its actions. Initially, the LLM only decodes a thought r_t following $\pi_1(r_t|c_t)$. Subsequently, the LLM predicts an action a_t in a *separate context*, incorporating both the thought r_t and a set of simple rules $\mathcal{M}$ that determines permissible actions of this step. This is further guided by π_2, formulated as $a_t \sim \pi_2(a_t|r_t, \mathcal{M})$. $\mathcal{M}$ encapsulates the governing rules of the environment (*e.g.*, the relation argument for `get_neighbors()` must be derived from the output of `get_relations()`, which is applied to the specified entity argument in prior steps), infusing prior knowledge into the LLM's decision-making process (see Figure 5.2(b)).

5.4 Benchmarks

The predominant tasks for databases and KBs are text-to-SQL parsing and KBQA. However, *popular benchmarks for them may fall short for evaluating language agents out-of-box.* Specifically,

[22]For databases, we directly use the error message from sqlite3. For KBs, we manually define several simple templates for error feedback along with each tool.

Model	Req. Cont. (N)		Req. Cont. (Y)		Overall	
	EX	**VA**	**EX**	**VA**	**EX**	**VA**
w/ Oracle Knowledge						
API Docs Prompt [136]						
w/ GPT-3.5-turbo	38.1	78.4	32.1	74.6	36.1	77.2
w/ GPT-4	49.5	95.5	41.7	89.9	46.9	93.7
w/o Oracle Knowledge						
API Docs Prompt [136]						
w/ GPT-3.5-turbo†	30.9	82.9	10.9	80.0	24.4	82.0
w/ GPT-4	38.2	91.6	13.8	93.1	30.4	92.1
StructGPT [71]						
w/ GPT-3.5-turbo	36.2	86.5	8.7	80.8	27.3	84.7
w/ GPT-4	40.7	93.4	13.5	91.1	31.8	92.6
MIDDLEWARE (*error feedback*)						
w/ GPT-3.5-turbo	38.8	95.7	19.8	94.7	32.7	95.4
w/ GPT-4	45.1	98.8	38.3	97.2	42.9	98.3

Table 5.1: Results on BIRD's dev set. Performance of all baselines is obtained under a *zero-shot* setting. † denotes the best method *w/o* oracle knowledge on BIRD's official leaderboard. The predictions with API Docs Prompt are directly supplied by the authors of BIRD.

the majority of questions in popular KBQA datasets like WEBQSP [10, 195] are one-hop or two-hop questions, for which we can effectively handle with existing semantic parsing methods [171]. Additionally, the databases featured in SPIDER [200] and WIKISQL [209] have limited complexity in terms of both schema design and the number of rows in the tables. This over-simplification enables the direct feeding of the database schema to the LLM, achieving strong performance without the need to access the actual content of the database [136]. Therefore, we need different benchmarks with complex environments and instructions that better mirror the real-world situations language agents must handle.

Databases For databases, we leverage BIRD [86], which is a recent dataset notable for its complexity, featuring intricate instructions over highly complex databases. There are originally two

Model	Counting		Superlative		None		Overall	
	F1	VA	F1	VA	F1	VA	F1	VA
Pangu$^{\diamondsuit}$ [169]								
w/ GPT-3.5-turbo	10.1	100.0	9.0	100.0	23.4	100.0	18.1	100.0
w/ GPT-4	12.3	100.0	14.2	100.0	35.6	100.0	27.1	100.0
KB-Binder [88]								
w/ GPT-3.5-turbo (20-shot)	0.0	33.7	0.2	19.4	6.7	37.0	4.2	32.8
w/ GPT-4 (20-shot)	7.9	48.3	0.4	28.2	6.0	45.8	5.2	42.6
StructGPT [71]								
w/ GPT-3.5-turbo	4.5	50.6	3.9	51.5	11.4	57.1	8.6	54.8
w/ GPT-4	2.2	37.1	3.9	30.1	11.7	26.3	8.4	29.0
MIDDLEWARE (*error feedback*)								
w/ GPT-3.5-turbo	33.7	70.7	22.0	64.1	23.9	56.8	25.3	60.8
w/ GPT-4	70.7	96.6	39.9	74.5	55.8	74.0	55.1	78.0
MIDDLEWARE (*decoupled generation*)								
w/ GPT-3.5-turbo	48.9	97.7	29.5	88.0	32.1	77.3	34.3	83.0
w/ GPT-4	74.1	98.9	42.6	85.1	61.0	83.6	59.3	85.8

Table 5.2: Results on KBQA-AGENT. All models are provided with *one-shot* demonstration except for KB-Binder, where we provide 20-shot demonstrations for optimal performance. $\diamondsuit$ indicates our reimplementation of Pangu, as the original code lacks support for chat models. We assume perfect entity linking for all methods.

different settings in BIRD: with and without oracle knowledge, where the oracle knowledge supplies specific information about the target database needed to fulfill each task. For instance, *"Exclusively virtual refers to Virtual = 'F'"*. With such oracle knowledge, the complexity of the environments is substantially mitigated; it offers a shortcut for the task and eliminates the necessity for deep engagement with the database. This cheating setting is also unrealistic for practical applications. As a result, we stick to the setting without oracle knowledge. For each of the 1534 questions in BIRD's dev set, we manually label whether accessing the database content is necessary to compile the SQL queries, noting that access is unnecessary if all mentioned values in a question exactly match database cells. This facilitates decomposing the language agent's performance based on questions that require deeper database engagement (496 questions) versus not (1038 questions) and enables

fine-grained insights into the LLM's performance. In addition to execution accuracy (**EX**) used in BIRD, which determines if the execution results of the predicted SQL match those of the ground truth SQL, we also evaluate whether the predicted SQL is a valid SQL query (**VA**).

KBs We curate KBQA-AGENT, a new test set sourcing from existing KBQA datasets that contain complex questions. In particular, we selected 500 diverse questions that involve at least three relations, or two relations coupled with an aggregation function (*i.e.*, **Counting** or **Superlative**). For each question, we annotate it with a ground truth sequence of actions based on the toolset defined by us.[23] Specifically, KBQA-AGENT comprises questions from three KBQA datasets on FREEBASE: GRAILQA [170], COMPLEXWEBQ [168], and GRAPHQ [161], ensuring a wide range of question types and sources. KBQA-AGENT is designed to be more representative of challenging, real-world scenarios compared to existing benchmarks. It offers an ideal testbed for evaluating language agents in interacting with massive KBs. We assess this through two metrics: **F1** of answer entities and Validity (**VA**), a binary metric evaluating the LLM's ability to find an answer, whether correct or not.

5.5 Experiments

5.5.1 Setup

Implementation To concretely instantiate our tools for the two environments, we employ standard query interfaces for databases and KBs, specifically SQLite for databases and Virtuoso for KBs. We then prompt the LLM with the tool descriptions together with the input task instructions. Each environment exhibits its own unique characteristics and challenges. In KBQA, the arguments for each function are either a variable or an item from the KB schema (*i.e.*, a relation or an attribute). In contrast, in text-to-SQL parsing, the arguments can be more varied, ranging from a part of a SQL query to a complete query. This makes listing potential actions, as needed in *decoupled generation*,

[23] We leverage the gold S-expressions provided by [173].

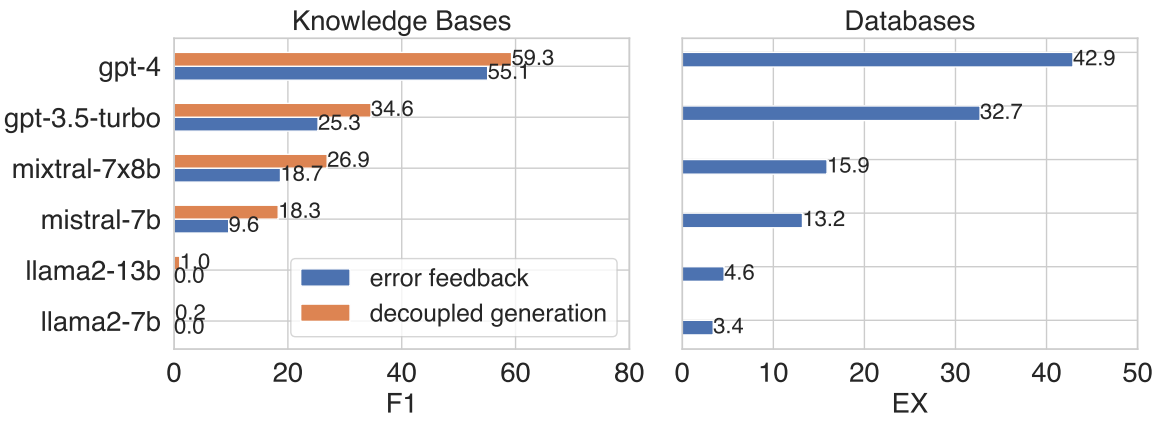

Figure 5.3: The open-source LLMs still largely lag behind GPT-3.5-turbo and GPT-4 in both environments.

much more complex for text-to-SQL parsing. Therefore, we implement *error feedback* solely for text-to-SQL parsing.

For the underlying LLMs, we primarily compare MIDDLEWARE with baseline methods using two of the most advanced LLMs—GPT-3.5-turbo-0613 [119] and GPT-4-0613 [118]—since our goal is investigating the full potential of tool-enhanced LLMs operating within complex environments. In addition, we also explore four open-source LLMs to more comprehensively evaluate our framework: Llama2-7B-Chat, Llama2-13B-Chat [175], Mistral-7B-Instruct-v0.2 [69], and Mixtral 8×7B-Instruct-v0.1 [70].

Baselines To fully understand the potential of tool augmentation for assisting LLMs in handling complex environments, we compare MIDDLEWARE against an array of strong baselines. For text-to-SQL parsing, LLMs demonstrate a strong ability to compose SQL queries when properly prompted with the database schema (*i.e.*, API docs prompting [136]). This also represents the current state-of-the-art prompting-based method when oracle knowledge is not available on BIRD's leaderboard. In adddition, we also compare with more baselines on BIRD's leaderboard that originally did not

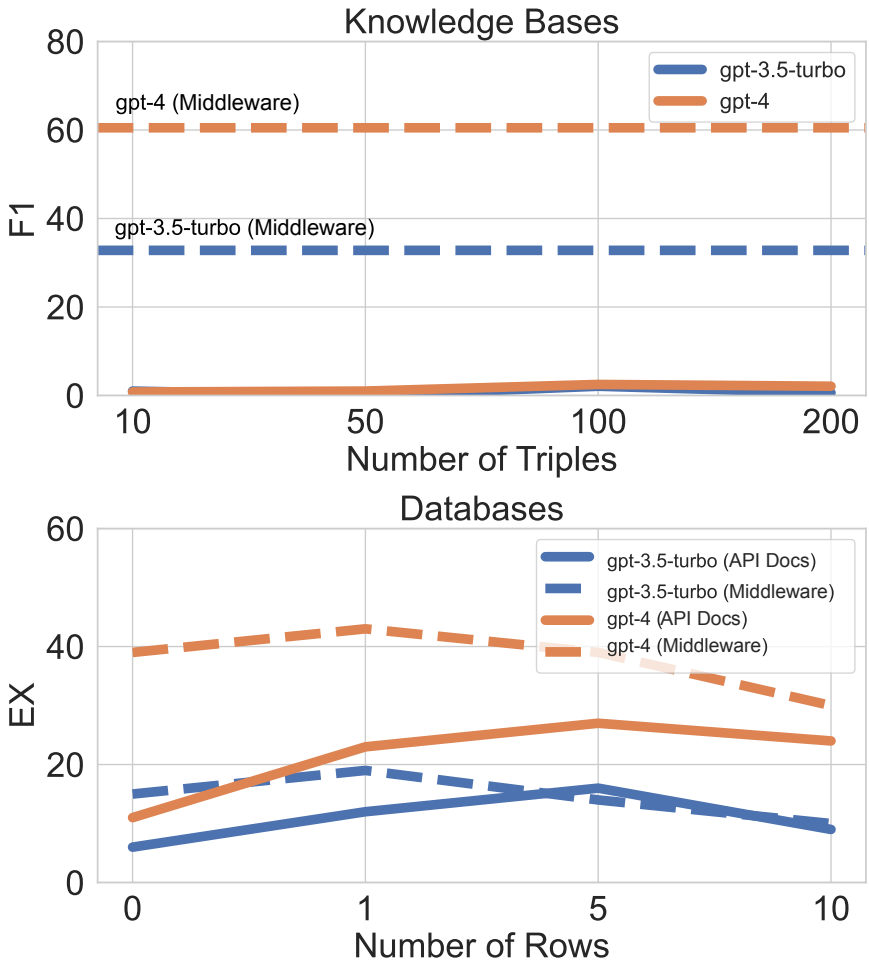

Figure 5.4: The customized tools can serve as effective *middleware* between the LLM and the environment.

submit their results using no oracle knowledge. For all methods on text-to-SQL parsing, we adopt the *zero-shot* setting. Unlike text-to-SQL parsing, directly prompting LLMs does not generate reasonable outputs for KBQA due to the massive size of the KB schema. Instead, existing KBQA methods based on LLMs typically follow two paradigms: either first generating an ungrounded program and then grounding the program to the KB schema afterwards [88, 115], or gradually constructing a complex program and grounding it step by step [169]. We compare MIDDLEWARE with the most representative work from each paradigm, namely KB-Binder [88] and Pangu [169]. We also include StructGPT as an additional baseline for tool use. For all KBQA methods except KB-Binder, we provide a *one-shot* demo to obtain more meaningful results.

5.5.2 Main Results

As shown in Tables 5.1 and 5.2, equipping LLMs with customized tools leads to significant improvement over previous standards, almost doubling or tripling the performance under multiple metrics. Specifically, API docs prompting can only feed the schema information to the LLM due to the vast amount of database content. As a result, it fails catastrophically on examples that require database content to compose the SQL query. In contrast, MIDDLEWARE equips the agent with tools to actively navigate the database to collect relevant information for composing a SQL query. As a result, MIDDLEWARE significantly closes the gap between performance on questions requiring database content and questions not requiring it when using GPT-4 (*i.e.*, 45.1% vs. 38.3%). Additionally, we notice that MIDDLEWARE minimizes the gap between with and without oracle knowledge from 15.5% to 4.0% using GPT-4 and 11.7% to 3.3% using GPT-3.5-turbo. Finally, StructGPT demonstrates a similar trend to API docs prompting because its tools do not provide any information about the database content. For KBQA, MIDDLEWARE demonstrates uniformly superior performance across different question types and significantly outperforms Pangu with both GPT-3.5-turbo and GPT-4. In particular, when equipped with GPT-4, MIDDLEWARE + *decoupled generation* outperforms Pangu

by 32.2% in F1. As for the other two baselines, KB-Binder and StructGPT, both fail miserably on our challenging setting. On the one hand, KB-Binder only retrieves relations within two hops from the entities for grounding. However, most questions in KBQA-AGENT involve more than two relations. As a result, many of its drafted programs are unable to ground, which explains its low VA. On the other hand, StructGPT is heavily limited by its constrained toolset and cannot handle complex questions in KBQA-AGENT. Therefore, StructGPT frequently refuses to provide an answer (as revealed by its low VA) due to insufficient information. The strong performance of MIDDLEWARE underscores that tools are instrumental for language agents in complex environments.

5.5.3 Experiments with Open-Source LLMs

To gain a more thorough insight, we also include experiments with four open-source LLMs (Figure 5.3). Our findings indicate that Llama2 models generally underperform compared to other LLMs, aligning with trends observed in other LLM leaderboards, such as Chatbot Arena [208]. Specifically, we find Llama2 models struggle with even generating grammatical tool use following our instruction. On the other hand, Mistral and Mixtral demonstrate much better performance than Llama2. In particular, Mixtral represents an advanced mixture-of-experts model that has demonstrated superior performance and even surpasses GPT-3.5-turbo on Chatbot Arena [208]. However, different from answering open-ended questions, properly engaging with the complex environment demands the language agent to produce more precise actions that strictly conform to the task specification. There is still a gap between Mixtral and GPT-3.5-turbo in terms of predicting valid actions over complex environments. Compared to GPT-3.5-turbo, Mixtral tends to output invalid actions more frequently. This also explains why *decoupled generation*, where the output space is strictly constrained to a list of valid actions, helps weaker models more. With MIDDLEWARE + *decoupled generation*, using Mistral can almost match the best baseline performance with GPT-3.5-turbo, and using Mixtral can

even match the best baseline with GPT-4. While stronger models like GPT-4 can effectively recover the mistake via *error feedback*, weaker models tend to benefit more from *decoupled generation*.

5.5.4 Tools as A Middleware Layer

To deepen our understanding of the integral roles of tools in aiding LLMs in accessing complex environments (*i.e.*, KB triples and database rows in our setup), we conduct further analysis by comparing MIDDLEWARE with prompting baselines with different amounts of data items directly sampled from the environment (Figure 5.4). For the KB, we sample 10, 50, 100, and 200 triples from FREEBASE based on the three-hop neighborhood of each entity in a question. These triples are the top-ranked ones using a sentence-BERT retriever [138] based on their similarity with the input question. We prompt the LLM directly with these sampled triples and request it to generate an answer to the given question. Given the extensive size of FREEBASE, accurately representing the environment with a mere subset of samples proves to be exceedingly difficult. Consequently, both GPT-3.5 Turbo and GPT-4 consistently yield an F1 score close to 0. For the database, we similarly augment API docs prompting with 1, 5, and 10 sampled rows for each table and evaluate on 100 random questions from BIRD that require accessing database content. Additionally, we also augment MIDDLEWARE with the same sampled rows in the database setting. We observe that including more database rows initially boosts baseline performance but eventually decreases it. With MIDDLEWARE, prompting the LLM with sampled rows yields minimal gain, and the standard setting without sampled rows already significantly outperforms all baselines. These results further confirm that the LLM, when augmented with tools, can effectively engage with complex environments, flexibly gathering the necessary information on demand and bypassing the limitations on the amount of data it can handle (*e.g.*, around 200 triples or 10 rows per table).

5.6 Conclusion

This chapter investigates the role of language prior in forming policies for various goals in natural language. To enable agents to better handle open-world tasks by interacting with complex real-world environments, we first craft a set of tools acting as *middleware* between LLMs and complex environments. The LLM-based agents can then determine how to chain these tools through multi-step decision making to accomplish goals. Experimental results show that with language-conditioned policy, the agent can largely outperform the method introduced in Chapter 4, when no training data is used. However, calling tools in such a *reactive* manner may still be limited when we need to perform search to better explore the environment. In the next chapter, we will introduce a method that further leverages the language prior for search (*i.e.*, model-based planning).

Limitations

In this chapter, we aim to address the compelling question we posed: how effectively can LLMs handle complex environments with the aid of tools? We investigate this through evaluations in two exemplary environments: KBs and databases. While we achieve notable results in these environments, it is important to acknowledge that implementing customized tools for KBs and databases presents fewer challenges compared to environments without a straightforward query interface, such as a webpage or a physical environment. In future work, we plan to extend MIDDLEWARE across a broader range of environments, aiming to fully realize the potential of language agents in complex environments through the integration of customized middleware tools.

Furthermore, the tools developed in this study are soley grounded in our experience. Despite this, our results already demonstrate the significant potential of augmenting LLMs with customized tools in complex environments, aligning with the primary objective of this chapter. Nonetheless, to enhance performance further, adopting a more principled strategy for tool design is essential. Additionally,

investigating autonomous tool-making methods [183] in complex environments presents a promising direction for future research.

Chapter 6: Enhance f with Model-Based Planning Conditioned on Natural Language Simulation

In the previous chapter, we have discussed how we can leverage LLMs' language proficiency to directly enhance the policy conditioned on natural language thoughts. In this chapter, we focus on another critical aspect of constructing the mapping function f: *search*. Performing search in open-world environments can be highly challenging to do the costly interaction and state-changing actions that prohibit *backtracking*. To perform search without the costly (and potentially dangerous) interactions with the environment, we need to have a computational representation of the environment, *i.e.*, a *world model*. To achieve this, we investigate how we can leverage LLMs to serve as such a world model by predicting the outcome of taking a specific action in natural language descriptions. Specifically, we propose WEBDREAMER, the first model-based planning framework that demonstrates promising results in the complex web environment, which features more costly interactions and state-changing actions compared with the KB and database environments. Our empirical results show that the LLM-simulated world models allow the agent to effectively conduct search within simulation and achieve better efficiency and safety compared with direct search within a browser.

6.1 Introduction

Planning [103]—the strategic search for optimal action sequences to achieve goals from initial states—has been fundamental to artificial intelligence since its inception, driving remarkable breakthroughs including superhuman performance in games like Go [153, 152]. Recent advances have demonstrated that integrating large language models (LLMs) with advanced planning algorithms (*e.g.*, [190, 57, 169, 180, 44, 17]) substantially enhances their performance on complex reasoning tasks beyond chain-of-thought (CoT) [184] approaches, with OpenAI's o1 [121] serving as a prominent example. These methods effectively scale inference-time compute and enable LLMs to explore multiple potential solution paths, which ultimately lead to more accurate outcomes.

Alongside these developments, research into generalist web agents capable of planning and executing a sequence of actions to complete complex tasks across diverse websites has garnered significant interest [39, 210, 207, 76], partly due to the web's potential as a complex yet realistic environment for driving agent research and development. However, applying existing planning algorithms to the online web environment presents formidable challenges. Chief among these challenges are the inherent *safety risks* associated with live website interactions [94], such as inadvertently submitting forms with sensitive information or triggering unintended transactions. These risks become even more pronounced when employing tree search algorithms [77, 132], as their exhaustive exploration can expose the agent to hidden vulnerabilities and unforeseen scenarios. Additionally, many online actions, such as confirming a purchase or sending an email, are *irreversible*, which further makes *backtracking*—a crucial component of planning algorithms—highly challenging, if not infeasible.

One promising solution to address these challenges is *model-based planning* [127, 110], which equips agents with the ability to simulate interactions using a *world model*—a computational representation of environment dynamics. By simulating action sequences within this virtual environment,

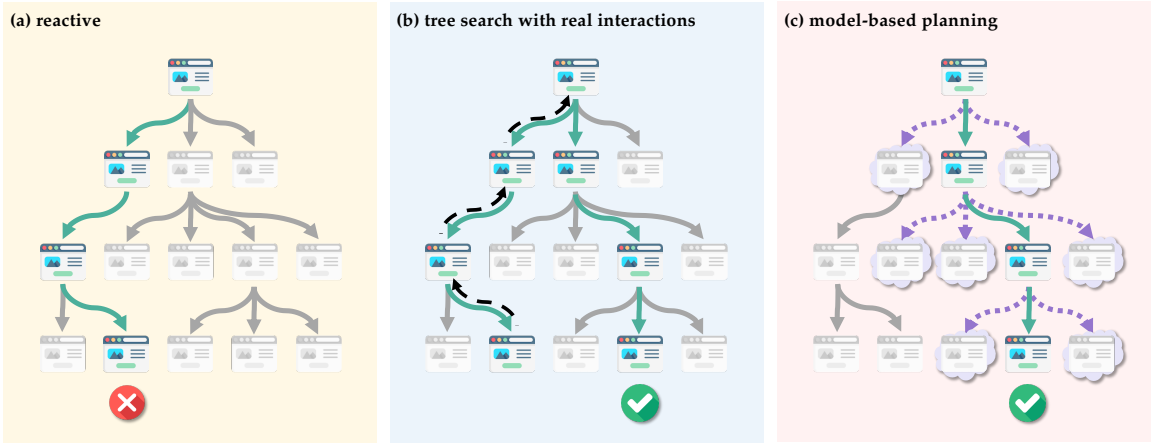

Figure 6.1: Schematic illustration of different strategies for web agents formulated as a search problem. Each node represents a webpage. (a) Reactive: The agent selects locally optimal actions without forward planning, often leading to suboptimal outcomes. (b) Tree search with real interactions: The agent explores multiple paths through active website navigation and permits backtracking (indicated by dashed arrows). However, in real-world websites, backtracking is often infeasible due to the prevalence of irreversible actions. (c) Model-based planning: The agent simulates potential outcomes (illustrated by cloud-bordered nodes) to determine optimal actions prior to real-world execution, thus minimizing actual website interactions while maintaining effectiveness. For visual clarity, only one-step simulated outcomes are depicted. Faded nodes indicate unexplored webpages, while green checkmarks and red crosses denote successful and unsuccessful outcomes, respectively. In this example, the tree search method visits 7 web pages over 9 steps of interactions, including backtracking, while model-based planning (virtually) explores 11 pages in just 3 steps, even with a simulation horizon of just 1.

agents can explore potential outcomes safely, without directly interacting with live websites. This approach not only reduces safety risks but also preserves the agent's capacity to explore and plan. Yet, the true challenge lies in creating a versatile world model that can faithfully capture the landscape of the ever-evolving Internet. While previous research demonstrates that LLMs can function as effective world models in simplistic settings like blocksworld [57] and gridworld [74], a bolder question emerges: *Can LLMs rise to the challenge of modeling the vast, dynamic Internet?* With their extensive pre-trained knowledge—spanning web structures, protocols, and user behaviors—LLMs are uniquely positioned to take on this task. Building on these insights, we present WEBDREAMER,

a pioneering framework that leverages LLMs as world models to navigate the web (Figure 6.1). At the core of WEBDREAMER lies the concept of “dreaming": before committing to any action, the agent uses the LLM to imagine the outcome of each possible step, expressed as natural language descriptions of how the state would change. These simulated outcomes are then evaluated based on their progress toward achieving the task objective. The most promising action is executed, and the process is repeated iteratively until the LLM determines that the goal has been reached (Section 6.4).

To validate the effectiveness of WEBDREAMER, we evaluate it on two representative benchmarks that support online interaction: VisualWebArena [76] and Mind2Web-live [126]. WEBDREAMER achieves substantial performance gains over reactive agents on both benchmarks, underscoring its practical value despite its conceptual simplicity. While tree search with actual interactions shows slightly superior performance on VisualWebArena, which features a controlled environment of three locally hosted websites, this method is rarely feasible in practical applications, given its inherent limitations regarding safety risks and the potential for irreversible actions in real-world websites. In contrast, our simulation-based approach offers a more flexible solution, balancing performance gains with practical applicability in real-world web navigation tasks.

In summary, our work introduces a new direction for AI planning in complex, real-world environments like the web using world models simulated by LLMs. With WEBDREAMER, we tackle the dual challenges of safety and complexity in web navigation. Our results validate the potential of LLM-based world models for planning in complex web environments and highlight new opportunities for optimizing LLMs as world models and improving model-based planning algorithms for language agents.

6.2 Related Work

6.2.1 Web Agents

Driven by the goal of automating tedious and repetitive web-based tasks, web agents powered by (multimodal) language models have made substantial progress in various aspects. Benchmarks have evolved from MiniWoB++ [147, 95] to WebShop [189] and WebArena [210], offering increasingly realistic website simulations. VisualWebArena [76] and Mind2Web [39] challenge models' ability to handle visual information and generalize across diverse tasks, websites, and domains.

Reactive Agents. Reactive agents make decisions based on immediate observations from the environment without performing any search or simulation of future actions, typically implemented with the ReAct framework [191]. Much progress has been made to enhance the fundamental capabilities of reactive web agents through both prompting closed-source models [207, 59, 39] and training models using HTML and webpage screenshots [84, 53, 46, 63, 7]. Additionally, models' abilities to ground web agent actions to elements have been improved through training on action-coordinate pair data [196, 31]. Further advancements have been achieved by training on web agent trajectories, utilizing both human-annotated trajectories [146, 63, 39, 79] and synthesized exploration trajectories [46, 157, 128]. However, reactive agents inherently suffer from short-sightedness, which can often lead to suboptimal performance in multi-step decision making.

Agents with tree search. Pan *et al.*[125] introduces a reward model based on GPT-4V, designed to provide both step-wise and trajectory-level rewards to guide inference-time search. Search Agent [77] investigates inference-time search algorithms in interactive web environments, enabling explicit exploration and multi-step planning. In contrast to Search Agent, which employs a variant of best-first tree search, AgentQ [132] and WebPilot [205] utilize Monte Carlo Tree Search (MCTS) as their primary search strategy.

While tree search on websites has demonstrated significant improvements, it still presents several limitations. First, the search process substantially increases inference time due to the need for extensive exploration, which is difficult to parallelize given its inherently sequential nature. Backtracking to previous states is essential for search-based methods but impractical on real-world websites. Koh *et al.*[77] addressed this in sandbox environments by storing action sequences to resume states after resetting the environment. However, resetting the environment or undoing action sequences is not feasible on live websites. Finally, the extra explorations introduced by search algorithms substantially amplify the risk of destructive actions that may irreversibly alter the website's state, potentially causing harmful side effects.

6.2.2 World Models

World models, a cornerstone of model-based reinforcement learning [110] since the introduction of Dyna [166], are typically trained on observed state transitions to predict future states and rewards. These world models enable efficient training through simulated experiences, reducing environmental interactions and improving sample efficiency [56]. Beyond their role in training, researchers have explored the use of world models to facilitate planning [127, 143]. Fundamentally, world models in reinforcement learning often involve task-specific training, with a primary focus on enhancing data efficiency in the agent learning process.

In contrast to traditional world models in reinforcement learning, LLMs employed as world models primarily focus on facilitating decision-making in planning rather than training. This distinction leads LLM-based models to prioritize key task abstractions over the high-fidelity simulations typically required in reinforcement learning. Recent research has demonstrated the potential of LLMs as world models for simple environments, leveraging their encoded broad world knowledge [57, 74]. Our study aims to advance this field by investigating the capabilities of LLM-based world models in more complex real-world environments, specifically diverse websites. A concurrent work [24] also

Action Type a	**Description**
click [`elem`]	Click on `elem`.
hover [`elem`]	Hover over `elem`.
type [`elem`] [`text`]	Type `text` into `elem`.
press [`key_comb`]	Press a key combo.
goto [`url`]	Go to `url`.
go_back	Click back.
go_forward	Click forward.
new_tab	Open a new tab.
tab_focus [`index`]	Focus on the i-th tab.
tab_close	Close current tab.
scroll [`up`/`down`]	Scroll up or down.
stop [`answer`]	End with an output.

Table 6.1: Action space for web navigation defined in VisualWebArena [76].

explores augmenting web agents with LLM-simulated action outcomes, however, their focus is on data collection to train an open-weights LLM, while ours centers on understanding the potential of this new paradigm using advanced LLMs such as GPT-4o [120].

6.3 Preliminary

6.3.1 Task Formulation

Web agents tasked with automating activities in live websites confront vast and complex search spaces. Formally, each task with a task instruction I can be framed as a partially observable Markov decision process (POMDP): $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \Omega)$, where $\mathcal{S}$ represents the set of all possible states of the environment, $\mathcal{A}$ represents all possible actions the agent can take, $\mathcal{O}$ represents the set of possible observations from the environment, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the state transition function, R is a binary reward denoting whether the task specified in I has been completed or not, and $\Omega : \mathcal{S} \to \mathcal{O}$ is a deterministic function that projects a state to an observation. The goal of the task is to execute a sequence of actions that achieves a reward of 1.

In practical scenarios, the environment is *partially observable* due to the complexity of web environments. The true state encompasses server-side variables, dynamically loaded content, hidden UI elements, and is subject to network conditions and browser limitations. Consequently, the agent can only perceive the environment through a limited viewport (*i.e.*, an observation $o \in \mathcal{O}$), which represents an incomplete projection of the true system state. The observation space typically manifests as screenshots or text-based accessibility trees, reflecting common implementation practices. This constrained observability naturally shapes the action space A, which comprises operations executable on interactable elements within o, such as element clicks, text input, and URL navigation (Table 6.1).

6.3.2 Planning through Simulation

Planning an optimal action sequence through tree search using real interactions governed by T is costly and risks irreversible actions. Model-based planning addresses these challenges by using a computational representation of the environment to simulate interaction outcomes. Instead of executing actions in the real environment, the agent leverages an approximate model to predict state transitions, enabling efficient exploration and evaluation of action sequences without real-world interactions. While *offline planning* can compute entire action sequences before execution in deterministic environments like BlocksWorld [57], web environments are too complex for such long-term prediction. This necessitates *online planning* approaches that interleave planning and execution, computing one action at a time.

One prominent approach is Model Predictive Control (MPC; Garcia *et al.* [48]), which iteratively simulates future trajectories to select actions. At each state s, MPC simulates trajectories over a finite horizon H for each possible action $a \in \mathcal{A}$ using a simulator function $\texttt{sim}(s, a)$ and evaluates them using a scoring function $\text{score}(\tau)$.[24] The action leading to the most promising trajectory is

[24]For example, $H = 1$ means simulating the outcome of a. $H = 2$ means simulating the outcome of a, then taking one additional step to propose a new action based on that simulated outcome, followed by simulating the outcome of this new action as well.

then executed: $a^* = \text{argmax}_{a \in \mathcal{A}} \texttt{score}(\texttt{sim}(s, a))$. This process repeats after observing the new state, allowing the agent to adapt its plan based on actual outcomes while avoiding costly real-world exploration. In reality, we cannot access the real state due to partial observability, as a result, we instead do $\texttt{sim}(o, a)$ using the observation $o = \Omega(s)$.

6.4 WEBDREAMER: Model-Based Planning for Web Agents

In this chapter, we propose WEBDREAMER, a pioneering approach leveraging LLMs as world models to enable efficient planning in complex digital environments. Our approach is motivated by the observation that web interfaces, despite their complexity, are designed to be predictable for human users. When browsing websites, humans can effectively anticipate action outcomes based on visual cues and common design patterns—clicking a “Submit" button leads to form submission, selecting a product image navigates to its detail page. Given that LLMs are trained on vast amounts of web-related data, we hypothesize that they have acquired sufficient knowledge to simulate the consequences of user actions, potentially serving as effective world models for planning.

6.4.1 Core Design

WEBDREAMER follows the planning through simulation paradigm introduced in Section 6.3.2. Figure 6.2 illustrates this process with three candidate actions, where WEBDREAMER simulates two-step trajectories for each action, selects the trajectory with the highest score, and executes its corresponding initial action. At its core, WEBDREAMER leverages an LLM to implement both the simulation function `sim` and the scoring function `score`.

Implementation for `sim`: Our implementation of `sim` consists of two modules: one predicts state changes after action execution, approximating T, while the other imagines a possible action based on the predicted state. Together, these two modules generate trajectories of length H, where H is a configurable horizon parameter (*i.e.*, the simulation depth). Specifically, to represent the state

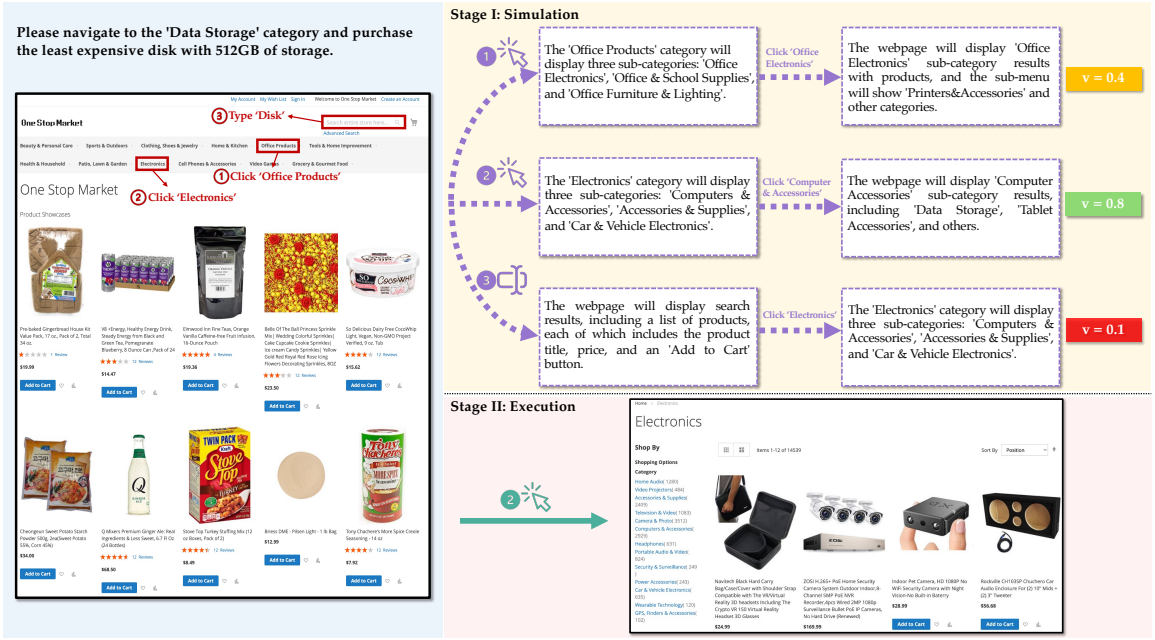

Figure 6.2: Illustration of WEBDREAMER using the LLM to simulate the outcome of each candidate action. The LLM simulates trajectories in natural language descriptions for three candidate actions: (1) *Click "Office Products"*, (2) *Click "Electronics"*, and (3) *Type "Disk" into textbox*. Through these simulations, each resulting trajectory is scored to identify the action most likely to succeed. In this case, the LLM selects Click *Click "Electronics"* as the optimal step and executes it. Each dotted box represents an LLM-generated state description after each simulated action. This example demonstrates a two-step planning horizon.

changes, we prompt the LLM to generate a concise natural language description focusing only on the effects of the action. For example, in Figure 6.2, the LLM would output a short description as follows when prompted to predict the effect of executing the action *Click "Electronics"*:

> → Click "Electronics"
>
> The 'Electronics' category will display three sub-categories: 'Computers & Accessories', 'Accessories & Supplies', and 'Car & Vehicle Electronics'.

Based on this predicted state, the LLM then imagines the next action (*i.e.*, *Click "Computers & Accessories"*), which leads to another state change prediction. This process generates a trajectory of horizon $H = 2$.

Implementation for `score`: After collecting a trajectory τ_i simulated from each candidate action a_i using `sim`, we further use the LLM as a scoring function for each simulation. Following [77], we prompt the LLM to evaluate each simulated trajectory with a three-scale response—complete (1.0), on track (0.5), or incorrect (0)—indicating its progress toward task completion. The final score is computed by averaging multiple samples of these evaluations.

Algorithm 2: WEBDREAMER

Input: Instruction I; initial observation o_0
Output: Sequence of actions $a_0, a_1, \ldots, a_T$

$t \leftarrow 0$;
while *True* **do**
  $\mathcal{A}_t \leftarrow \texttt{get\_candidate}(I, o_t)$;
  $\mathcal{A}'_t \leftarrow \texttt{self\_refine}(\mathcal{A}_t)$;
  $a_t = \text{argmax}_{a \in \mathcal{A}'_t} \texttt{score}(\texttt{sim}(o_t, a))$;
  $o_{t+1} \leftarrow \texttt{execute}(a_t)$;
  $t \leftarrow t + 1$;
  if `termination_check()` = *True* **then**
    break;
  end
end

In addition to `sim` and `score`, a prerequisite to planning is candidate action generation. We employ a two-stage approach: first sampling top-k actions following [77], then using LLM to self-refine unnecessary actions for simulation. This self-refinement step is motivated by our observation that at different steps, the same k can introduce varying degrees of irrelevant actions—some steps naturally have fewer plausible actions than others. We show the pseudo code of WEBDREAMER's overall design in Algorithm 2. `termination_check` verifies if the model outputs a `stop` action, reaches max steps, or repeats an action over 3 times, also following the implementation by [77].

6.4.2 Discussion

To justify our design choices in light of our goal—a pioneering study on using LLMs as world models for web environments—we discuss three key considerations:

State change description instead of HTML. While we use natural language descriptions to capture state changes, an alternative is to prompt the LLM to predict the HTML or accessibility tree of the resulting page. However, since most webpage elements remain unchanged after an action, predicting the entire page structure is unnecessarily wasteful. Moreover, such concrete predictions are more prone to hallucination—HTML requires precise details about the website, whereas state descriptions need only capture the essential changes. For our pioneering study, we embrace this simpler, more intuitive representation, though we make no claims about its strict superiority over HTML or accessibility trees (see Section 6.6.1 for a detailed analysis).

Prompting instead of fine-tuning. In this work, we implement WEBDREAMER through direct prompting of state-of-the-art LLMs (*i.e.*, GPT-4o [120]) without fine-tuning. Our rationale is straightforward: we aim to first establish the feasibility of using advanced LLMs as world models for web environments and their effectiveness in planning. Demonstrating promising results with this

approach will lay the foundation for future work on optimizing this direction through fine-tuning OSS models on targeted datasets.

MPC-based planning instead of MCTS. We adopt a relatively straightforward MPC-based planning algorithm rather than more sophisticated approaches like MCTS that have been prominent in recent LLM planning research [57, 44]. This choice is motivated by our empirical findings: increasing the planning horizon of WEBDREAMER yields diminishing returns, which suggests the current limitations of LLMs in accurately modeling multi-step trajectories (see Section 6.6.1). Given our goal of exploring LLMs as world models for web environments, this simpler approach suffices to demonstrate the key insights while acknowledging the current capabilities of LLMs.

6.5 Experiments

6.5.1 Setup

To properly test our planning framework's real-world performance, we use benchmarks with online evaluation, capturing the dynamic nature of web interactions. We focus on two representative benchmarks: VisualWebArena (VWA; [76]), which emphasizes a multimodal setting, and Mind2Web-live [126], which operates with HTML by default. VWA comprises 910 tasks across three locally hosted websites: Shopping, Classifieds, and Reddit. In contrast, Mind2Web-live includes 104 tasks spanning 69 real-world websites. We adhere to the default settings of both benchmarks: for VWA, we use screenshots with Set-of-Marks prompting as the observation space, while for Mind2Web-live, we use HTML. For our LLM, we choose the most advanced multimodal LLM available, GPT-4o, as it best serves our aim to pioneer model-based planning with LLMs and explore the full potential of this envisioned paradigm. In our experiments, we empirically set the planning horizon H to 1.[25]

[25]We find $H = 2$ works slightly better than $H = 1$ in our setting, but due to the significantly higher cost, we opt for $H = 1$. A comprehensive analysis of this parameter is presented in Section 6.6.1.

To demonstrate the effectiveness of our proposal, we primarily compare our approach with two major baselines: the reactive agent and the tree search agent with real interactions.[26] While we can readily implement our own method for both benchmarks, for the tree search baseline [77], we can only compare with it on VWA, because of the infeasibility of doing tree search in real-world websites in Mind2Web-live. Specifically, in VWA, Koh*et al.* [77] keeps track of the sequences of actions to get to states in previous trajectories. During backtracking, they reset the sandbox and re-execute the action sequence to restore the state. However, resetting the environment to undo effects is not always feasible in real-world websites featured in Mind2Web-live.

6.5.2 Main Results

Effectiveness. We present the overall performance results in Table 6.2. WEBDREAMER demonstrates substantial improvements over the reactive agent on both VWA and Mind2Web-live datasets. Notably, on the VWA dataset, our proposed method achieves a 33.3% relative performance gain. Meanwhile, our proposal still underperforms the tree search baseline in terms of overall success rate. Although our approach still falls short of the tree search baseline in terms of overall success rate, it is crucial to emphasize that tree search is not a practical option for real-world websites, whereas WEBDREAMER provides a more flexible and adaptive alternative. On Mind2Web-live, WEBDREAMER outperforms the reactive baseline by 2.9% (a relative gain of 13.1%), which is less significant than the improvement on VWA. However, it is worth noting that the Mind2Web-live dataset does not offer as much discriminative power, as evidenced by the minimal performance differences across multiple base LLMs shown in Table 6.2. The strong results on both VWA and Mind2Web-live indicate the effectiveness of our method across different observation settings.

[26]We will refer tree search with real interactions simply as tree search in our experiments for brevity.

Benchmark	Observation $\mathcal{O}$	Method	Completion Rate	Success Rate
VisualWebArena	Screenshot+SoM	Gemini-1.5-Pro + Reactive [76]	-	12.0%
		GPT-4 + Reactive [76]	-	16.4%
		GPT-4o + Reactive [76]	-	17.7%†
		GPT-4o + Tree Search [77]	-	26.4%
		GPT-4o + WEBDREAMER	-	23.6% (↑33.3%)
Mind2Web-live	HTML	GPT-4 + Reactive [126]	48.8%	23.1%
		Claude-3-Sonnet + Reactive [126]	47.9%	22.1%
		Gemini-1.5-Pro + Reactive [126]	44.6%	22.3%
		GPT-4-turbo + Reactive [126]	44.3%	21.1%
		GPT-3.5-turbo + Reactive [126]	40.2%	16.5%
		GPT-4o + Reactive [126]	47.6%	22.1%
		GPT-4o + WEBDREAMER	49.9%	25.0% (↑13.1%)

Table 6.2: Results on VisualWebArena and Mind2Web-live. WEBDREAMER significantly outperforms the reactive baseline and falls only slightly short of the tree search baseline on VWA while requiring far fewer website interactions. For Mind2Web-live, implementing tree search algorithms poses significant challenges due to the requirement for website backtracing, leading us to omit tree search performance metrics. This limitation further underscores the flexibility of our model-based planning method. We also include additional baselines (denoted by gray cells) to provide broader context. While these comparisons may not directly assess our core hypothesis, they offer valuable background for understanding our method's performance in the web navigation landscape. † We run the reactive baseline on VWA by ourselves because local hosting requirements may lead to hardware-dependent performance variations.

We further conduct a more granular analysis comparing our proposed method to the reactive baseline on the VWA dataset across multiple dimensions. Table 6.3 demonstrates that our model-based planning approach consistently outperforms the reactive baseline across all websites and task difficulty levels. On tasks of medium difficulty according to the official annotation by VWA, model-based planning even surpasses the performance of tree search (*i.e.*, 22.2% vs. 24.1%). Despite its promise, model-based planning still struggles with hard tasks in VWA that necessitate multistep simulations. The accuracy of simulations diminishes as the number of steps increases, presenting a significant challenge for handling hard tasks.

Websites	Reactive	Tree Search	WEBDREAMER	γ
Classifieds	16.8%	26.5%	22.6%	59.8%
Reddit	15.3%	20.5%	18.6%	63.5%
Shopping	19.4%	29.0%	26.5%	74.0%

(a) Websites

Difficulty	Reactive	Tree Search	WEBDREAMER	γ
Easy	28.8%	42.3%	37.4%	63.7%
Medium	16.4%	22.2%	24.1%	132.8%
Hard	10.7%	14.9%	12.7%	47.6%

(b) Task Difficulty

Table 6.3: Success rate breakdown based on different dimensions. $\gamma = \frac{SR_{\text{WEBDREAMER}} - SR_{\text{reactive}}}{SR_{\text{tree search}} - SR_{\text{reactive}}}$ measures the extent to which WEBDREAMER narrows the gap between the reactive agent and the tree search agent.

Efficiency. Another key advantage of model-based planning is its efficiency compared with tree search using actual explorations. As shown in Table 6.4, tree search requires approximately three times more steps than the baseline across all environments, whereas our method maintains comparable action steps. Notably, tree search introduces about ten times more wall clock latency due to the extra actions and backtracking, while the simulation overhead in our approach is minimal and can be further reduced with increased parallelization.

Steps	Reactive	Tree Search	WEBDREAMER
Classifieds	3.4	9.9	4.1
Reddit	5.1	13.6	5.2
Shopping	4.5	11.4	4.5

(a) Number of Action Steps

Seconds	Reactive	Tree Search	WEBDREAMER
Classifieds	68.3	749.2	183.6
Reddit	83.5	972.1	233.7
Shopping	87.7	785.7	179.4

(b) Task Completion Wall Clock Time

Table 6.4: Action steps and wall clock time on VWA.

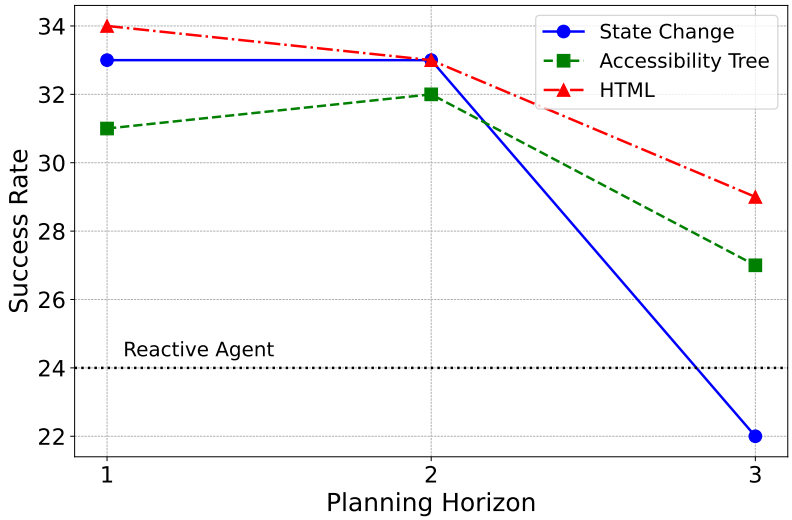

Figure 6.3: We demonstrate the performance on a subset of the VWA dataset, varying both the state representation within simulations and the planning horizon. Planning with long horizon with simulation remains challenging, regardless of the state representation employed.

6.6 Analyses

6.6.1 State Representation And Planning Horizon

Our model-based planning approach relies on two dimensions for simulation: the state representation and the planning horizon H (*i.e.*, the simulation depth). To gain deeper insights into its effectiveness and limitations of exisiting LLMs, we investigate how various configurations affect the final performance. Given the high computational cost of these experiments, we conduct this analysis using a subset of the VWA dataset, comprising 100 shopping tasks with officially annotated human trajectories.

In addition to the state change description used in our primary experiments, we explore alternative approaches where GPT-4o predicts either the HTML code or the accessibility tree of the resulting webpage within the simulation. For each of these state representations, we evaluate planning horizons of 1, 2, and 3 steps. As depicted in Figure 6.3, all three representations consistently outperform the

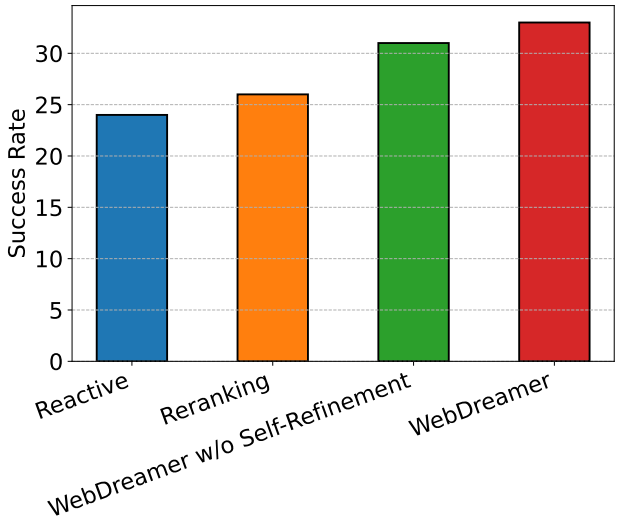

Figure 6.4: Ablation study on the simulation stage and self-refinement stage.

reactive baseline across all horizon settings, except for state change description with $H = 3$. This underscores the robustness of our proposal to use LLMs as world models for the web environment and highlights the LLM's general understanding of the environment regardless of the output format. Nonetheless, when the planning horizon extends to 3 steps, the effectiveness across all settings begins to diminish, particularly for the state change description. Upon closer examination, this is primarily due to action proposal hallucinations within the simulation. Specifically, the action proposal within simulation is biased toward generating seemingly relevant actions for task completion, even when these actions are not available based on the predicted outcome. As a result, as the planning horizon increases, the trajectories simulated from different actions become less distinguishable, as they all appear somewhat correct. However, achieving better performance with longer horizons is not a major goal of this work, instead, we aim to show the feasibility and potential of using LLMs for model-based planning. Also, our findings may inform future efforts to further improve the model-based planning framework.

6.6.2 Ablation Study

To determine if the observed improvements come from specific parts of our model-based planning approach, we perform ablation studies on the simulation and self-refinement stages, using the same subset from Section 6.6.1. We pay special attention to the simulation stage, which is the core of model-based planning. One might argue that the primary improvement stems from reranking candidate actions, irrespective of whether this ranking relies on simulation. To test this idea, we conduct an experiment where we remove the simulation stage completely and instead ask the reward model to directly evaluate each candidate action. As shown in Figure 6.4, this modified reranking approach does lead to some improvement over the ractive baseline, but the gain is small and still falls well behind WEBDREAMER. These results confirm that the LLM-based world model simulation plays a crucial role in the planning process. Furthermore, we observe a decrease in performance when removing the self-refinement stage. Upon closer examination, we find that this decline is primarily due to the self-refinement module's ability to effectively filter out less relevant candidate actions when the next optimal action is clear. In contrast, directly simulating all actions may introduce additional noise that can negatively impact performance.

6.6.3 Case Study

To clarify the role of simulation in planning, we present a case study covering both positive and negative examples. This illustrates how simulation aids the agent in exploring the environment, as well as how inaccuracies in simulation can lead to incorrect predictions. Specifically, an error case caused by imperfect world model simulation is shown in Figure 6.5, while a positive case where the simulation leads to correct action prediction is shown in Figure 6.6.

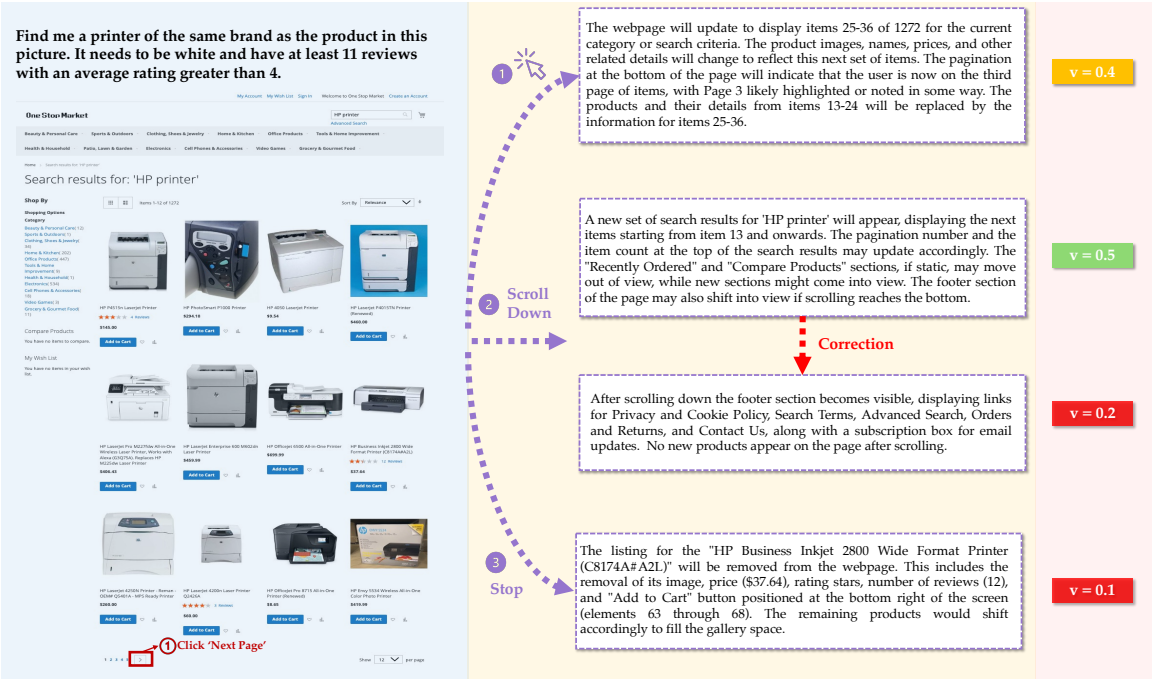

Figure 6.5: An error case caused by imperfect world model simulation.

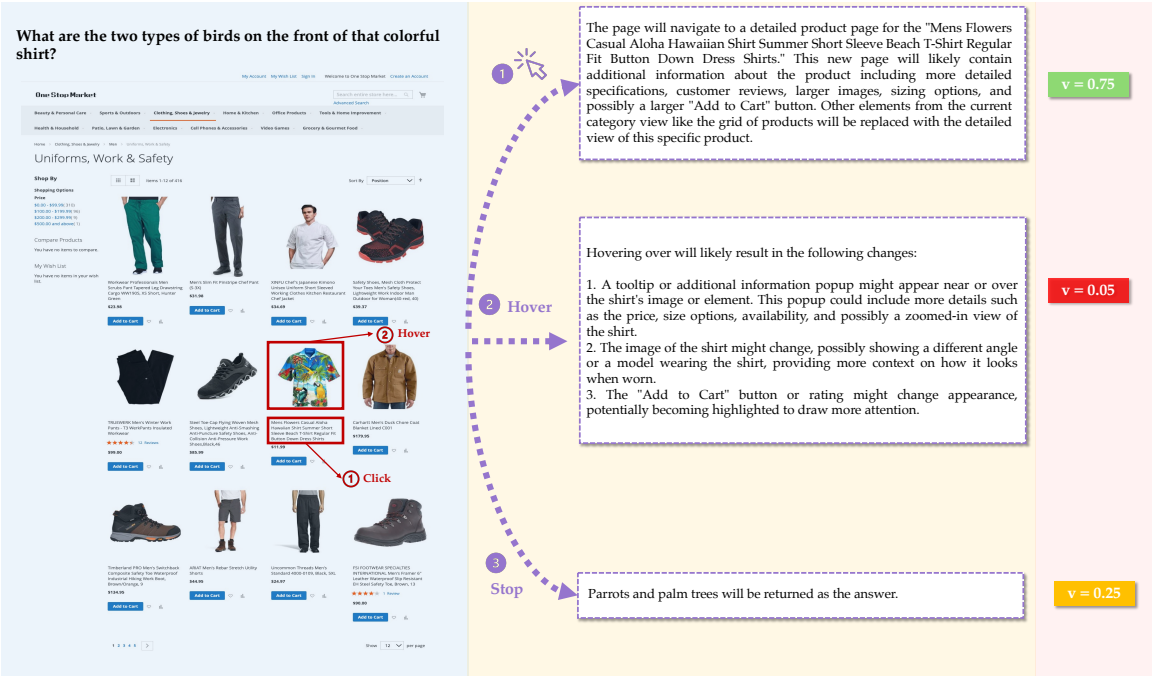

Figure 6.6: A positive case where the simulation leads to correct action prediction.

6.7 Conclusion

In this chapter, we demonstrate the strong potential of using LLMs as world models to support planning in complex environments, which further facilitate the construction of f. Specifically, our model-based planning approach, WEBDREAMER, shows substantial improvement over reactive baselines and offers greater flexibility than tree search, which is often impossible in real-world websites. As a pioneering effort in this area, our work opens new avenues for model-based planning with LLM-simulated world models. Future work can focus on further optimizing LLMs as world models for complex environments and developing more robust model-based planning algorithms for long-horizon planning in extrapolating agents to various goals.

Chapter 7: Conclusion and Future Prospects

Developing AI agents that can extrapolate to novel goals in open-world environments remains one of the most challenging and fundamental pursuits in AI research. In this thesis, we have systematically formulated the problem of extrapolating AI agents to diverse open-world tasks through explicit goal modeling (see Figure 1.2 for a general intuition). Specifically, there are two core components for extrapolation: a model $\mathcal{M}: G \to \mathcal{Z}$ that can preserve the structure of goals in a latent space and a mapping function $f: G \to \Pi$ that can map the goals to the corresponding policies. Grounded in our definition of $\mathcal{M}$ and f, we have discussed numerous efforts to implement extrapolative agents, spanning from systematic evaluation of extrapolation to advanced methods with large language models (LLMs).

The core thesis statement is that *we can leverage the natural language capacity of modern language models to better extrapolate agents to diverse goals in open-world tasks*. Specifically, we focus on two aspects: 1) using language models to provide a solution for $\mathcal{M}$ and 2) using language models to provide a prior to building the mapping f, including both direct policy and search. The intuition behind modeling goals in natural language is that natural language inherently entails a concept space which allows similarity measurement and composing new concepts via other concepts. On the one hand, such a concept space is a key ingredient for agents to extrapolate to novel goals that correspond to new concepts in the space. On the other hand, modern language models can effectively derive a concept space through pre-training on massive corpora, and we can leverage

language models as the backbone to ground policies corresponding to various goals in such a latent concept space.

To systematically evaluate the problem of extrapolation instead of asking ad-hoc questions like "*if an agent knows how to water flowers in kitchen and fold clothes in bedroom, can it also water flowers in bedroom?*," we qualitatively define three types of extrapolation based on the relationship between new goals and seed goals using the KBQA environment (Chapter 2). Through our evaluation, we have gained numerous key insights, including the critical role of language models in non-i.i.d. extrapolation and the superiority of discrimination and constrained decoding serving as f. These insights lead to two methods that achieve strong extrapolation performance on KBQA: a unified framework for constrained decoding with pre-trained language models (Chapter 3) and a general framework that is compatible with any type of language model for discrimination (Chapter 4).

In addition, recent advances in LLMs have demonstrated great promise in using natural language prior as a reasoning tool, which allows the model to effectively handle diverse tasks with the broad knowledge encoded in LLMs. This thesis also investigates how to better leverage natural language in the planning stage to better extrapolate AI agents to new goals. Specifically, we have addressed both the two key ingredients of planning (*i.e.*, policy and search): We show that language-conditioned policy can help agents to better extrapolate compared with direct action prediction (Chapter 5). Additionally, we also investigate the use of LLMs as world models for the complex web environment (Chapter 6). For the first time, we demonstrate that by simulating action outcomes through natural language descriptions, we can effectively conduct search within these simulated environments, yielding promising results.

Through these studies, this thesis provides a basis for studying extrapolative agents in open-world tasks. However, it is just a beginning. There are at least three interesting future directions to further pursue this fundamental goal in AI research:

1. **Advanced design choice of the mapping function.** Throughout this thesis, we have focused on using the same mapping function f for all goals within a single agent. However, this may not be sufficient in some cases. Sepcifically, considering the example shown in Figure 7.1, when we instruct OpenAI Operator [122], which is the state-of-the-art web agent, to "*Find origami tutorials for each of the 12 animals in Chinese Zodiac*"—a compositional goal that can be decomposed into twelve sub-goals—Operator fails miserably by only finding the tutorial for two animals. Nonetheless, if we manually decompose the goal into sub-goals, Operator can actually achieves each sub-goal perfectly. This indicates the deficiency of using the mapping function f for all goals in current agents. Future work may either consider modeling the policy of compositional goals through explicit task decomposition or leverage the decomposed policies to bootstrap f during training to better handle novel goals.

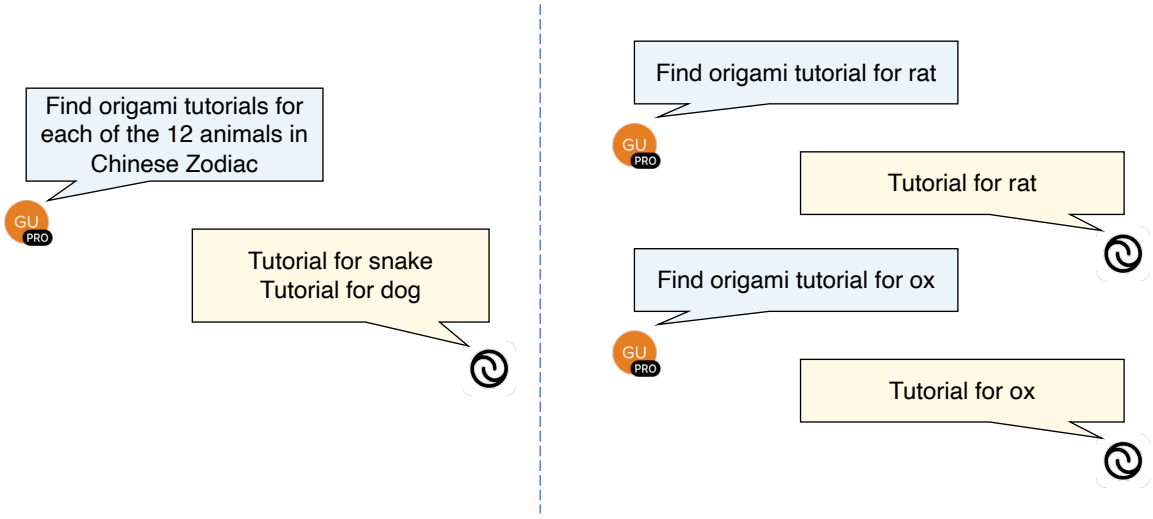

Figure 7.1: OpenAI Operator knows how to achieve each sub-goal of "*Find origami tutorials for each of the 12 animals in Chinese Zodiac*" (*right side*), while still fails to accompolish the compositional goal per se (*left side*). This may indicate the deficiency of using the same mapping function f for all goals.

2. **Extrapolation with minimal natural language prior.** When natural language prior can be used to facilitate the construction of f, we should make full use of it. Natural language prior is usually more helpful in high-level planning like decomposing *"make a pizza"* into a list of sub-goals. However, it can be less helpful for low-level policies. For example, the procedural and commonsense knowledge encoded in natural language may have limited utility in scenarios such as low-level robot control, which requires precise output of control arguments for various degrees of freedom [158]. Though a promising method is to perform hierarchical planning by decomposing the policy into plans in high-level robotic control language [104, 156] and low-level motion planning (*i.e.*, task-motion planning [72]), a more ambitious possibility is to achieve the goal through end-to-end methods. In this case, the challenge lies in constructing the mapping function f with minimal reliance on natural language priors. Effectively extrapolating AI agents in such scenarios demands more advanced modeling of the policy space without the language modality (*e.g.*, through representation learning of low-level robotic skills), as well as improved techniques for bridging the often disjointed goal and policy spaces. Progress in this direction is crucial for overcoming the critical sample efficiency bottleneck that currently limits the practical deployment of VLA models [75].

3. **Concept space without natural language modeling.** While we advocate for the critical role of natural language in developing extrapolative agents for open-world tasks, we do not claim that the use of natural language is *necessary*. In essence, what we seek is a robust model of the goal space that captures the necessary properties for effective extrapolation, as discussed in Chapter 1, which does not inherently depend on natural language. An intuitive example is that animals like cats or dogs, despite not understanding natural language, also demonstrate certain extrapolative behaviors. Ultimately, what we want is a concept space that enables agents to ground different policies, perform abstraction, and make analogies. Such a concept space represents, arguably, the ultimate goal in AI research [62]. The reason we currently rely on natural language for extrapolative agents is that

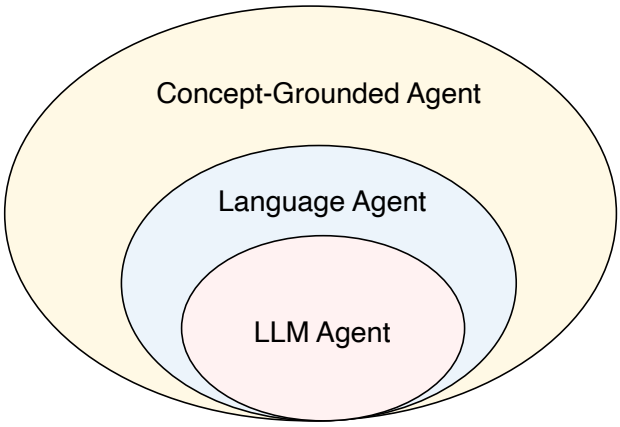

Figure 7.2: Different levels to look at extrapolative agents. Currently, we rely on natural language modeling to build extrapolative agents. The intuition is that learning from natural language serves as a shortcut to derive a concept space for modeling goals. In the long term, however, we anticipate the emergence of concept-grounded agents that may evolve independently of natural language learning.

learning from natural language serves as a shortcut to (approximately) achieve such a concept space, and viable alternatives remain elusive. In the future, however, we envision concept-grounded agents that may or may not form their conceptual foundations based on natural language (Figure 7.2).

Research on AI agents has long been considered the primary domain of the reinforcement learning (RL) community. The advances in LLMs have enabled more researchers from natural language processing (NLP) backgrounds to enter agent research. However, there remains consistent controversy and criticism regarding the role of LLMs in agent research, particularly from the RL community. In this thesis, we aim to justify the role of natural language in agent research. Specifically, agents enhanced with natural language do not aim to replace RL agents; rather, they can serve as a complementary component that boosts the extrapolation capabilities of RL-trained agents while potentially addressing the challenges of data efficiency inherent in traditional reinforcement learning approaches. Finally, we hope this thesis helps agent researchers with backgrounds in NLP better

understand their role and responsibility in agent research, and inspires them to join forces with researchers across diverse specializations to collaboratively advance the frontier of agent research.

Bibliography

[1] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th international conference on world wide web*, pages 1191–1200, 2017.

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

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[4] 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. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571, 2020.

[5] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[6] 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. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

[7] Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. Screenai: A vision-language model for ui and infographics understanding. *ArXiv preprint*, abs/2402.04615, 2024.

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

[9] Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1431–1440, 2015.

[10] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 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, October 2013. Association for Computational Linguistics.

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

[12] Nikita Bhutani, Xinyi Zheng, and HV Jagadish. Learning to answer complex questions over knowledge bases with query composition. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 739–748, 2019.

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

[14] Ben Bogin, Shivanshu Gupta, and Jonathan Berant. Unobserved local structures make compositional generalization hard. *arXiv preprint arXiv:2201.05899*, 2022.

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

[16] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015.

[17] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *ArXiv preprint*, abs/2407.21787, 2024.

[18] Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

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

[20] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[21] Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6101–6119, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[22] Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 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, May 2022. Association for Computational Linguistics.

[23] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In Zoubin Ghahramani, editor, *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, 2007.

[24] Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. *arXiv preprint arXiv:2410.13232*, 2024.

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

[26] Khyathi Raghavi Chandu, Yonatan Bisk, and Alan W. Black. Grounding 'grounding' in NLP. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4283–4305. Association for Computational Linguistics, 2021.

[27] Shuang Chen, Qian Liu, Zhiwei Yu, Chin-Yew Lin, Jian-Guang Lou, and Feng Jiang. 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, August 2021. Association for Computational Linguistics.

[28] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *CoRR*, abs/2304.05128, 2023.

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

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

[31] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[32] 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. Binding language models in symbolic languages. *CoRR*, abs/2210.02875, 2022.

[33] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 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, October 2014. Association for Computational Linguistics.

[34] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. Kbqa: learning question answering over qa corpora and knowledge bases. *arXiv preprint arXiv:1903.02419*, 2019.

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

[36] Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 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, November 2021. Association for Computational Linguistics.

[37] Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer, 2015.

[38] Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 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, June 2021. Association for Computational Linguistics.

[39] Xiang Deng, **Yu Gu**, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

[40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics.

[41] Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. Towards a question answering system over the semantic web. *Semantic Web*, 2020.

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

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

[44] Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *ArXiv preprint*, abs/2309.17179, 2023.

[45] Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*, 2020.

[46] Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. *ArXiv preprint*, abs/2305.11854, 2023.

[47] Evgenly Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, version 1, 2013.

[48] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

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

[50] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: large language models can self-correct with tool-interactive critiquing. *CoRR*, abs/2305.11738, 2023.

[51] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *CoRR*, abs/2305.14909, 2023.

[52] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy, July 2019. Association for Computational Linguistics.

[53] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *ArXiv preprint*, abs/2307.12856, 2023.

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

[55] Bernal Jiménez Gutiérrez, Yiheng Shu, **Yu Gu**, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[56] David Ha and Jürgen Schmidhuber. World models. *ArXiv preprint*, abs/1803.10122, 2018.

[57] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[58] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *CoRR*, abs/2305.11554, 2023.

[59] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *ArXiv preprint*, abs/2401.13919, 2024.

[60] Jonathan Herzig and Jonathan Berant. Don't paraphrase, detect! rapid and effective data collection for semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3810–3820, 2019.

[61] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[62] Douglas R Hofstadter and Emmanuel Sander. *Surfaces and essences: Analogy as the fuel and fire of thinking*. Basic books, 2013.

[63] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.

[64] Yuncheng Hua, Yuan-Fang Li, Guilin Qi, Wei Wu, Jingyao Zhang, and Daiqing Qi. Less is more: Data-efficient complex question answering over knowledge bases. In *Journal of Web Semantics*, 2020.

[65] Zixian Huang, Ao Wu, Jiaying Zhou, **Yu Gu**, Yue Zhao, and Gong Cheng. Clues before answers: Generation-enhanced multiple-choice qa. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3272–3287, 2022.

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

[67] Naman Jain, Skanda Vaidyanath, Arun Shankar Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram K. Rajamani, and Rahul Sharma. 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, 2022.

[68] Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 12–22, Berlin, Germany, August 2016. Association for Computational Linguistics.

[69] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

[70] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, 2024.

[71] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. StructGPT: A general framework for large language model to reason over structured data. *CoRR*, abs/2305.09645, 2023.

[72] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[73] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*, 2019.

[74] Doyoung Kim, Jongwon Lee, Jinho Park, and Minjoon Seo. Cognitive map for language models: Optimal planning via verbally representing the world model. *ArXiv preprint*, abs/2406.15275, 2024.

[75] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024.

[76] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *ArXiv preprint*, abs/2401.13649, 2024.

[77] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.

[78] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[79] Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5295–5306, 2024.

[80] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR, 2018.

[81] Yunshi Lan and Jing Jiang. 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, July 2020. Association for Computational Linguistics.

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

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

[84] Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 18893–18912. PMLR, 2023.

[85] Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 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, November 2020. Association for Computational Linguistics.

[86] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls. *CoRR*, abs/2305.03111, 2023.

[87] Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A benchmark for tool-augmented llms. *CoRR*, abs/2304.08244, 2023.

[88] Tianle Li, Xueguang Ma, Alex Zhuang, **Yu Gu**, Yu Su, and Wenhu Chen. Few-shot in-context learning on knowledge base question answering. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[89] 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. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[90] Zhenyu Li, Sunqi Fan, **Yu Gu**, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18608–18616, 2024.

[91] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 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, July 2017. Association for Computational Linguistics.

[92] Percy Liang. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*, 2013.

[93] 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. Holistic evaluation of language models. *CoRR*, abs/2211.09110, 2022.

[94] Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. EIA: environmental injection attack on generalist web agents for privacy leakage. *CoRR*, abs/2409.11295, 2024.

[95] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *6th International Conference*

on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[96] Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

[97] Xiao Liu*, Hao Yu*, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, **Yu Gu**, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*, 2024.

[98] Xiao Liu*, Tianjie Zhang*, **Yu Gu***, Iat Long Iong, Xixuan Song, Yifan Xu, Shudan Zhang, Hanyu Lai, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. Visualagentbench: Towards large multimodal models as visual foundation agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[100] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. *CoRR*, abs/2304.09842, 2023.

[101] Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical*

Methods in Natural Language Processing, pages 2185–2194, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[102] Gaurav Maheshwari, Priyansh Trivedi, Denis Lukovnikov, Nilesh Chakraborty, Asja Fischer, and Jens Lehmann. Learning to rank query graphs for complex question answering over knowledge graphs. In *ISWC*, 2019.

[103] Marcelo G Mattar and Máté Lengyel. Planning in the brain. *Neuron*, 110(6):914–934, 2022.

[104] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Experimental robotics: the 13th international symposium on experimental robotics*, pages 403–415. Springer, 2013.

[105] John McDermott. R1 ("xcon") at age 12: lessons from an elementary school achiever. *Artificial Intelligence*, 59(1-2):241–247, 1993.

[106] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *CoRR*, abs/2302.07842, 2023.

[107] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 2013.

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

[109] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[110] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[111] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020.

[112] 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. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332, 2021.

[113] Allen Newell, John C Shaw, and Herbert A Simon. Report on a general problem solving program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA, 1959.

[114] Ngonga Ngomo. 9th challenge on question answering over linked data (qald-9). *language*, 2018.

[115] Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. Code-style in-context learning for knowledge-based question answering. *CoRR*, abs/2309.04695, 2023.

[116] OpenAI. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[117] OpenAI. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[118] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[119] OpenAI. Models - OpenAI API. https://platform.openai.com/docs/models/gpt-3-5, 2023.

[120] OpenAI. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/, 2024. Accessed: 2024-09-28.

[121] OpenAI. Introducing OpenAI o1. https://openai.com/o1/, 2024. Accessed: 2024-09-29.

[122] OpenAI. Introducing operator. https://openai.com/index/introducing-operator/, 2025. Accessed: April 12, 2025.

[123] Vardaan Pahuja, **Yu Gu**, Wenhu Chen, Mehdi Bahrami, Lei Liu, Wei-Peng Chen, and Yu Su. A systematic investigation of kb-text embedding alignment at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1764–1774, 2021.

[124] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, 2009.

[125] Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024.

[126] Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. Webcanvas: Benchmarking web agents in online environments. *ArXiv preprint*, abs/2406.12373, 2024.

[127] Razvan Pascanu, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sebastien Racanière, David Reichert, Théophane Weber, Daan Wierstra, and Peter Battaglia. Learning model-based planning from scratch. *ArXiv preprint*, abs/1707.06170, 2017.

[128] Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. Large language models can self-improve at web agent tasks. *ArXiv preprint*, abs/2405.20309, 2024.

[129] Xutan Peng, Yipeng Zhang, Jingfeng Yang, and Mark Stevenson. On the security vulnerabilities of text-to-sql models. *arXiv preprint arXiv:2211.15363*, 2022.

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

[131] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[132] Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *ArXiv preprint*, abs/2408.07199, 2024.

[133] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining

Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models. *CoRR*, abs/2304.08354, 2023.

[134] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789, 2023.

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

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

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

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

[139] Ohad Rubin and Jonathan Berant. 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, June 2021. Association for Computational Linguistics.

[140] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 07–09 Jul 2015. PMLR.

[141] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761, 2023.

[142] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 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, November 2021. Association for Computational Linguistics.

[143] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[144] Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein,

Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571, September 2020.

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

[146] Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina Toutanova. From pixels to UI actions: Learning to follow instructions via graphical user interfaces. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[147] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144. PMLR, 2017.

[148] Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 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, November 2021. Association for Computational Linguistics.

[149] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 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, 2020.

[150] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

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

[152] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[153] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

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

[155] Chan Hee Song, Jihyung Kil, Tai-Yu Pan, Brian M Sadler, Wei-Lun Chao, and Yu Su. 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, 2022.

[156] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. LLM-Planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

[157] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *ArXiv preprint*, abs/2403.02502, 2024.

[158] Mark W Spong and Mathukumalli Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008.

[159] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

[160] Yu Su. Language agents: a critical evolutionary step of artificial intelligence. *yusu.substack.com*, Sep 2023.

[161] Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 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, November 2016. Association for Computational Linguistics.

[162] Yu Su and Xifeng Yan. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1235–1246, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[163] Danny Sullivan. A reintroduction to our Knowledge Graph and knowledge panels. `https://blog.google/products/search/about-knowledge-graph-and-knowledge-panels/`, 2020.

[164] Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. Sparqa: skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8952–8959, 2020.

[165] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[166] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[167] Chang-Yu Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. Exploring chain of thought style prompting for text-to-sql. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5376–5393. Association for Computational Linguistics, 2023.

[168] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[169] **Yu Gu**, Xiang Deng, and Yu Su. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4928–4949. Association for Computational Linguistics, 2023.

[170] **Yu Gu**, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond I.I.D.: three levels of generalization for question answering on knowledge bases. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3477–3488. ACM / IW3C2, 2021.

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

[172] **Yu Gu**, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. Middleware for llms: Tools are instrumental for language agents in complex environments. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7646–7663, 2024.

[173] **Yu Gu** and Yu Su. 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, October 2022. International Committee on Computational Linguistics.

[174] **Yu Gu***, Kai Zhang*, Yuting Ning*, Boyuan Zheng*, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *CoRR*, abs/2411.06559, 2025.

[175] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[176] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In *The Semantic Web–ISWC*

2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16, pages 210–218. Springer, 2017.

[177] Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond Mooney, Trevor Darrell, and Kate Saenko. Captioning images with diverse objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5753–5761, 2017.

[178] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 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, July 2020. Association for Computational Linguistics.

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

[180] Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves llm search for code generation. *ArXiv preprint*, abs/2409.03733, 2024.

[181] Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 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, November 2021. Association for Computational Linguistics.

[182] Yushi Wang, Jonathan Berant, and Percy Liang. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*

and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1332–1342, Beijing, China, July 2015. Association for Computational Linguistics.

[183] Zhiruo Wang, Daniel Fried, and Graham Neubig. Trove: Inducing verifiable and efficient toolboxes for solving programmatic tasks. *CoRR*, abs/2401.12869, 2024.

[184] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in neural information processing systems*, 2022.

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

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

[187] Bowen Xu. What do you mean by "open world"? In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

[188] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3795–3802. IEEE, 2018.

[189] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information*

Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022.

[190] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[191] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[192] Xuchen Yao. Lean question answering over Freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 66–70, Denver, Colorado, June 2015. Association for Computational Linguistics.

[193] Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 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, May 2022. Association for Computational Linguistics.

[194] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 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, July 2015. Association for Computational Linguistics.

[195] Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 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, August 2016. Association for Computational Linguistics.

[196] Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. *ArXiv preprint*, abs/2404.05719, 2024.

[197] Donghan Yu, **Yu Gu**, Chenyan Xiong, and Yiming Yang. Compleqa: Benchmarking the impacts of knowledge graph completion methods on question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12748–12755, 2023.

[198] Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*, 2022.

[199] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 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, 2021.

[200] 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. 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, October-November 2018. Association for Computational Linguistics.

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

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

[203] Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. Complex question decomposition for semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4477–4486, Florence, Italy, July 2019. Association for Computational Linguistics.

[204] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. Editing-based SQL query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China, November 2019. Association for Computational Linguistics.

[205] Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. *ArXiv preprint*, abs/2408.15978, 2024.

[206] Boyuan Zheng*, Michael Y. Fatemi*, Xiaolong Jin*, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, **Yu Gu**, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *CoRR*, abs/2504.07079, 2025.

[207] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024.

[208] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, 2023.

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

[210] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2023.

[211] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning T5 for text ranking with ranking losses. *CoRR*, abs/2210.10634, 2022.